SPEECH UNDERSTANDING SYSTEMS

BOLT BERANEK AND NEWMAN, INCORPORATED

PREPARED FOR
OFFICE OF NAVAL RESEARCH

MAY 1976

BBN Report No. 3303

SPEECH UNDERSTANDING SYSTEMS

Quarterly Technical Progress Report No. 6

1 February 1976 to 30 April 1976

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 3303	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): SPEECH UNDERSTANDING SYSTEMS Quarterly Technical Progress Report No. 6 1 February 1976 to 30 April 1976	5. TYPE OF REPORT & PERIOD COVERED: Q.T.P.R. 2/1/76 - 4/30/76
	6. PERFORMING ORG. REPORT NUMBER: BBN Report No. 3303
7. AUTHOR(s): W. Woods, M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue.	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0533
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 5D30
11. CONTROLLING OFFICE NAME AND ADDRESS: ONR Department of the Navy Arlington, VA 22217	12. REPORT DATE: May, 1976
	13. NUMBER OF PAGES: 158
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Acoustic-phonetic experimental facility, acoustic-phonetics, acoustic-phonetic recognition, acoustic-phonetic rules, acoustics, dynamic programming, grammar complexity, HWIM, labeling, lattice filtering, lexical retrieval, multi-component systems, natural language retrieval system, natural language understanding, parsing, phonological rules, phonology,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes recent progress of the BBN Speech Understanding Systems project covering the period from February 1976 to April 1976. The BBN Speech Understanding project is an effort to develop a continuous speech understanding system which uses syntactic, semantic and pragmatic support from higher level linguistic knowledge sources to compensate for the inherent acoustic indeterminacies in continuous spoken utterances. These knowledge sources are integrated with sophisticated signal

(cont'd.)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

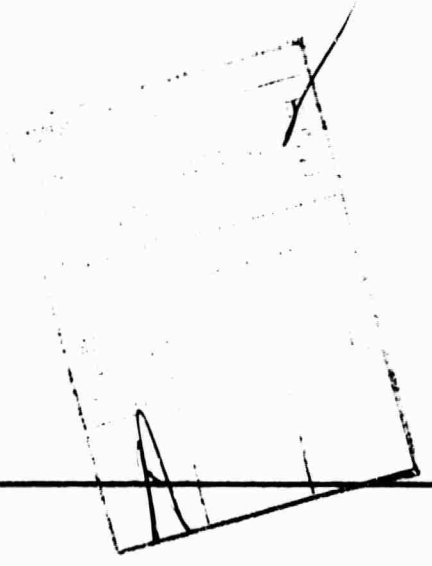
19. Key Words (cont'd.)

pragmatic grammar, progabilistic labeling, probabilistic lexical retrieval, question-answering, recognition strategies, resource allocation, segmentation, segment lattice, semantic network, shortfall algorithm, shortfall density, speech, speech data base, speech processing, speech recognition, speech understanding, SUR, synthesis-by-rule, system organization, system performance, word recognition, word spotting, word verification.

---

20. Abstract (cont'd.)

processing and acoustic-phonetic analysis of the input signal, to produce a total system for understanding continuous speech. The system contains components for signal analysis; acoustic parameter extraction; acoustic-phonetic analysis of the signal; phonological expansion of the lexicon; lexical matching and retrieval; syntactic, semantic, and pragmatic analysis and prediction; and inferential fact retrieval and question answering, as well as synthesized text or spoken output. This sixth quarterly technical progress report contains technical notes on probabilistic lexical retrieval, on a new recognition strategy based on "shortfall density," and on the evolving uses of knowledge in the system.



SPEECH UNDERSTANDING SYSTEMS
Quarterly Technical Progress Report No. 6
1 February 1976 to 30 April 1976

ARPA Order No. 2904

Program Code No. 5D30

Name of Contractor:
Bolt Beranek and Newman Inc.

Effective Date of Contract:
30 October 1974

Contract Expiration Date:
29 October 1976

Amount of Contract: $1,041,261

Contract No. N00014-75-C-0533

Principal Investigator:
William A. Woods
(617) 491-1850 x361

Scientific Officer:
Marvin Denicoff

Title:
SPEECH UNDERSTANDING SYSTEMS

QTPR Editor:
Bonnie Nash-Webber
(617) 491-1850 x227

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

Table of Contents

I. PROGRESS OVERVIEWS

A. Acoustic-Phonetic Recognition

Most of this past quarter has been spent in developing a flexible method for computing accurate label scores. In the current version of the Acoustic-Phonetic Recognition (APR) program, the score used by the word matcher when aligning a dictionary phoneme with a lattice segment is based on the probability of the particular confusion. As described in earlier reports [Schwartz and Zue, 1976], we would like to modify the probabilities for the more likely phonemes, based on the particular values of the acoustic parameters observed within the segment concerned.

In order to be able to do this, we have made the following additions to the Acoustic-Phonetic Experiment Facility (APEF) [Schwartz, 1976].

1) Display Filters

Once the user has specified a phonetic context and a set of acoustic features, and has scanned the data base, he can choose to display a smaller subset of the data by defining a filter. For instance, though the phonetic context for an experiment might include all plosives, he might want to display the labial plosives separately from the others. The filter allows this to be done without having to do the experiment twice.

2) Multi-Dimensional Distributions

The APEF now has the capability to compute non-parametric, multi-dimensional probability densities of up to 6 features. Given the names of the features and the segment of the phonetic context which is to be distinguished, the APEF automatically determines what the classes are and gathers a separate distribution for each class. Then it can test each sample point against each of the class distributions and compute the probability for each class. The APEF correctly eliminates the contribution of the test sample in the training set in order to ensure unbiased results. In this way, our entire data base can be used as both training and test sets while being able to estimate performance on new data.

3) APR Interface

Once a desirable set of features has been specified, the APEF can write all the necessary information on a file. There is also a function which the APR can call to read in the same file in order to use the probability density functions in scoring different labeling choices.

The APEF reports the results of the experiment in several ways. First, if the top scoring class is the wrong one, it writes the data and probabilities on a file, along with information about the exact time and utterance

involved. It records the ratio of the highest probability returned to the 2nd highest probability (if the highest was correct) or to the probability assigned to the correct class (if the highest was wrong). The logarithms of these ratios are computed, and two separate sums are maintained (correct and incorrect). We feel that these two sums are a much better measure of labeling performance than merely recording the percent correct decisions. The APEF also computes the average log of the ratios. This number is usually lower for incorrect decisions than for correct ones, indicating that the algorithm is rarely very sure of the result when it is wrong. Of course the number correct and incorrect are also reported.

We presented two papers on acoustic-phonetics at the ICASSP-IEEE conference held in Philadelphia, 12-14 April 1976. These were: "Acoustic-Phonetic Experiment Facility for the Study of Continuous Speech" [Schwartz, 1976], and "Acoustic-Phonetic Recognition in BBN SPEECHLIS" [Schwartz and Zue, 1976].

While preparing examples for the first paper, two new labeling algorithms were developed. The first enables us to distinguish correctly between [M,N,NX] 91% of the time, the second between [F,TH] 94% of the time. We are confident that these algorithms will improve the performance of the APR program.

B. Lexical Retrieval

During the last quarter, only relatively minor changes were made to the Lexical Retrieval component.

The first such change affected the handling of splits and merges across word boundaries. In the past, the Lexical Retrieval component accommodated for possible segmentation errors in two ways: by 'splitting' segments in which a boundary may have mistakenly been missed and by 'merging' adjacent segments between which a boundary may have mistakenly been inserted. While this permitted a 'correct' alignment of single words against the segment lattice, it did not permit joining words with segmentation errors postulated at their mutual boundary. In these cases a 'correct' mutual alignment could only be made if both phonemes (merge) or segments (split) matched at the mutual boundary were known. Since it is meaningless to retain this information for the alignment of single words, additional code was written which preserved it for use across word boundaries. After this, the entire speech system demonstrated a marked improvement in behavior. Sentences that had previously been blocked by a serious segmentation error at a word boundary were now successfully recognized.

The second change involved the interface between Lexical Retrieval and Control. In evaluating the system's behavior, it began to seem apparent that the system would

benefit by being able to dynamically vary certain parameters in the Lexical Retrieval component that had previously been defaulted by Control. Since the interface had not been designed to make varying these parameters easy, minor changes were made to it to facilitate more explicit control. In conjunction with this, a new parameter - the number of distinct fuzzy words to be returned - was defined and included in the new interface.

The third change involved the scoring of alignments where segmentation errors had been hypothesized, since it had previously been inconsistent with our scoring philosophy. Such consistency required that higher order probabilities be estimated. Specifically, P(Li|Pj Pk) (i.e., the probability of the phoneme sequence Pj Pk being labeled as Li) must be calculated for the scoring of a split, and P(Li Lj|Pk) (i.e., the probability of Pk being labeled as the segment sequence Li Lj) must be calculated for the scoring of a merge. The details of implementation of probabilistic split and merge were worked out this past quarter and some of the code written. We see the remainder of this work being completed in the near future.

C. Syntax

1, Grammar

This quarter the grammar was modified in minor ways in order to parse all 40 of the March and April sentences. Loops in the grammar that caused many spurious predictions were eliminated without reducing the number of natural sentences accepted by the grammar. In particular, a systematic effort was begun to "tighten up" the grammar so that few, if any, predictions would be made that are in some way incompatible with the theory causing the predictions. For example, in response to predictions from the grammar, a theory "... what trip for Bonnie July thirty to..." was created. This indicated that the point in the trip network that pushes for a date modifier should require a preposition as part of that modifier ("... trip for Bonnie on July thirty.."). Instances of such "funny" theories or events are now watched for in the traces produced by the speech system so that continuing modifications to the grammar can be made.

Also in this quarter, a package of LISP functions was written to measure the grammar's complexity in terms of its average branching ratio, that is, the average number of alternative paths that are possible at each step of the parse. The set of functions enable one to walk through the grammar counting the number of input-consuming paths for

input strings up to a specified word length. Branching ratios for the current grammar are shown in Table 1 as a function of sentence length in words.

These results are somewhat preliminary since further changes will be made in both the grammar and the measuring function. We hope to include in next quarter's report a technical note comparing BBN's SMALLGRAM to a grammar being used at IBM [Kahl et al., 1976] using this measure.

# Words in Sentence	# Possible Sentences	Av. Branching Ratio
1	11	11
2	207	14.388
3	15707	25.044
4	3.2655E5	23.903
5	7.3984E6	23.650
6	1.3806E8	22.734
7	2.5321E9	22.047
8	4.8292E10	21.615
9	8.8708E11	21.259
10	1.5425E13	20.836
11	2.5636E14	20.412
12	4.1661E15	20.028
13	6.7586E16	19.706
14	1.1064E18	19.447
15	1.8347E19	19.241
16	3.0657E20	19.072
17	5.1018E21	18.920
18	8.3630E22	18.770
19	1.3418E24	18.616
20	2.1027E25	18.456
21	3.2169E26	18.292
22	4.8033E27	18.124
23	6.9938E28	17.952
24	9.9181E29	17.777
25	1.3677E31	17.597

Table 1: SMALLGRAM measurements.

2) Parser

During the past quarter, development continued on our new parser. Most significantly, we eliminated a rare, but serious source of exponential explosion. The basic function of the parser is to find all possible paths through a given island, and if any, propose all words and categories possibly adjacent to it. An exponential explosion of possible paths could occur though when two adjacent words could be consumed several levels apart in the grammar. If the available context did not sufficiently constrain the paths through the intermediate levels, all possible paths, with all possible "empty" (i.e., non-consuming) intermediate levels would be enumerated. In some cases, a two- or three-word island would result in more than 400 paths being generated.

The new parsing algorithm allows the empty intermediate levels to be left out of the paths. The mechanism is similar to Earley's algorithm [Earley, 1970], broadened to work middle out. The grammar is indexed to create predictive sets that correspond to Earley's L* except that we must have such sets for four directions. We call these sets B! (Begin) for left-to-right top-down prediction (this corresponds exactly to L*), C! (Complete) for left-to-right bottom-up predictions, UC! (Un-Complete) for right-to-left top-down predictions and UB! (Un-Begin) for right-to-left

bottom-up predictions. We have chosen the "!" instead of the Kleene star for our notation since these sets include jump arcs in intermediate grammar levels as well as Push (Begin) and Pop (Complete) arcs.

If a path between two words (which we will think of as two consuming arcs) begins with a push arc, contains any number of intermediate push and jump arcs, and finally ends with a push arc, then we have a B! relationship between the two consuming arcs and we build a path with two segments (one for each consuming arc) to represent it. When predicting bottom-up, the parser must allow the possibility that the arc consuming the next word will be up one or more levels and then again down one or more levels into an adjacent constituent. The sequence of arcs that may be followed for making predictions becomes complicated in such cases, so we have devised a regular-expression like notation to represent the arc sequences that may be followed in generating the four predictive sets. In the notation we use J, B, and C to indicate Jump, Push and Pop arcs going left to right, and LJ, UB, and UC to represent the same arcs going right to left. Using this notation the predictive sets may be described as:

B! = B + ((J + B)* B)*

C! = C + ((J + C)* C)* + ((J + B)* B)*

UB! = UB + ((LJ + UB)* UB)* + ((LJ + UC)* UC)*

UC! = UC + ((LJ + UC)* UC)*

The parser has been made slightly faster, averaging 5 to 7 seconds for a syntactic event, as opposed to 12 seconds. The timings of the new parser are also more consistent than its predecessor. We are continuing to work on improving the efficiency of the parser and shaking out the remaining bugs.

D. The Semantic Network

As we have discussed in previous QTPR's, much of the semantic and world knowledge of HWIM is represented in a semantic network called "TRAVELNET." (The accessing and maintenance functions for the network make up the SEMNET system.) In this section we give an overview of the kinds of knowledge now embedded in TRAVELNET, as well as a brief discussion of how this knowledge is currently being used. The numbers given below are all approximate, since it is not in general possible to assign a single purpose to a node. There are currently 1465 nodes in the network.

1) Basic net maintenance knowledge - A substantial portion of the knowledge in TRAVELNET concerns how to represent and use other knowledge. This is the kind of knowledge that would be needed in even a "blank" network; i.e., one that had no word- or domain-specific knowledge. While much of our work up to this point has been on this basic level of knowledge, we do not expect it to expand

greatly in the future. Currently, about 22% of the network is used for basic knowledge, including the 271 link names that are now in use.

2) Knowledge of words and their use - A second major type of knowledge in TRAVELNET concerns words, how they are related and used. Part of this knowledge was the data base for the former Semantic Recognizer component that is no longer in use; e.g., case frames and word associations. Another part is knowledge of how to construct utterances for use in communicating with the travel budget manager. There are now 288 English words, of which 216 are base forms in TRAVELDICT. Word knowledge makes up about 17.5% of TRAVELNET.

3) World knowledge - Closely related to knowledge of words and language is knowledge of the world. This includes knowledge of people, places, and general facts, and has been organized in TRAVELNET in the form of an ISA hierarchy. World knowledge of this type is now being used by HWIM to augment the semantic and pragmatic knowledge that exists in SMALLGRAM. For instance, SMALLGRAM knows that a name can be made up of a first name, e.g., "Bill", a last name, e.g., "Bates", or a first name-last name pair, e.g., "Bill Bates". However, only TRAVELNET knows whether "Bill Bates" has a referent in HWIM's "world". Thus, our semantic and pragmatic knowledge is factored into one data structure

(SMALLGRAM) containing fairly constant information and a second (TRAVELNET) containing a more volatile sort. There are 26 people known to HWIM, 30 cities, 10 states and 10 countries. Together with other world knowledge, these facts make up 22% of TRAVELNET.

4) Specific factual knowledge - There is a class that might be called "factual knowledge" which consists of the details of specific projects, trips, budgets and conferences. It is distinguished from world knowledge in that it changes rapidly as events occur. In fact, whereas HWIM's world knowledge is now essentially fixed, its factual knowledge changes every time a trip is taken or money shifted from one budget item to another. Not surprisingly this is the part of TRAVELNET most likely to expand in the future. It is now about 32.5% of the network.

5) Computational knowledge - Knowledge of data base structure and computation is also represented in TRAVELNET. This knowledge exists in METHODS associated with particular link names [Bruce and Harris, 1975]. About 3% of the TRAVELNET nodes represent METHODS.

6) Discourse knowledge - There are two important kinds of discourse knowledge for HWIM. The first is knowledge of idealized discourse, what type of utterance is likely to be produced in a given state of the discourse. The second is knowledge of the current discourse state; e.g., what the

current topic is, what objects are available for anaphoric reference, who the speaker is and the current date. We plan to incorporate the latter knowledge soon into HWIM as a guide for relaxing constraints in the pragmatic grammar. For example, if a conference is under discussion, then "What is the registration fee?" is a meaningful utterance. If not, then the speaker can be expected to say, "What is the registration fee for the XXX conference?". Currently, about 3% of TRAVELNET represents idealized discourse knowledge.

TRAVELNET is realized physically in three files [Woods et al., 1975]. The only one loaded into core is a 2000 word index array in which each array location corresponds to a node. At each location there are either (1) file pointers into a second, randomly accessed file that contains the link-value pairs for the node, (2) pointers to list structure (in core) for the link-value pairs, or (3) node free list indicators and pointers. Access to a node may be by its number (the array index) or (for nodes with names) by its name. In the latter case a third file containing name-number pairs is searched for the correspondence. A flag can be set to direct whether file information, once accessed, is to remain in core. For the index file, 11 pages of storage are now used; for the list structure file, 58 pages; and for the name file, 5 pages. We expect these numbers to grow 10-25% during the remainder of the contract year. Fortunately, the maintenance of the network on

external files has eliminated a serious space problem we once had in the TRIP fork, without seriously impairing execution time. Furthermore, we normally run so that nodes once accessed remain in core so the cost of file access is paid only once.

E. Verification

During the past quarter, we have improved the speed of the Verification component by recoding its dynamic programming algorithm in PDP10 assembler language (macro). This has decreased the computation time required to verify a word by approximately 35%. In addition, we have added a dynamic error bound that terminates the parametric matching process if the distance measure exceeds a certain threshold. This error bound should cut down the time spent in verifying hypothesized words that are poor matches, thereby speeding up the verification process as a whole.

Up to now, scores returned by Verification have been incommensurate with those of Lexical Retrieval. To remedy this situation and enable the assignment of a single score based on both evaluation methods, we have collected Verification scores from ten sentences spoken by three speakers. The 300 words are used to compute separate score distributions for correct and incorrect words. From these distributions, we are now computing log-likelihood ratios

that will allow us to combine Verification match scores with those returned by the Lexical Retrieval component.

Our synthesis-by-rule program development has been adversely affected by the unavailability of our graphics display terminal and real-time interface due to the shift of hardware into our new computer rooms. The graphics terminal only became available toward the end of the quarter. We are now able to start work on improving the synthetic spectral model generated by the synthesis-by-rule program.

F. System Organization & Recognition Strategies

1) Recognition Strategies

The recognition strategies underwent two phases of change during the past quarter. The first phase involved a set of extensions to the control strategy used at the February 3 Dress Rehearsal Demonstration. The second phase constituted a major reworking of that strategy, involving both a change in scoring policy and other improvements.

Most of the extensions of the "island driven strategy" were directed at increasing the amount of parallelism, since the advent at the end of the last quarter of a faster and less space-bound Syntax fork had removed the imperative of recognizing an utterance in a small number of theories or not at all. These extensions included:

a) A change in shelving policy: Shelving of events for a father theory during a son's processing was changed to shelving only those events which would extend the father in the direction opposite to the current son. This prevented incompatible theories from being sprouted from opposite ends of a common father, yet was not so restrictive as shelving all of a father's events.

b) A new type of event: A "seed event" was included on the event queue for each unique word resulting from the initial scan. This meant that if the scores of evolving theories fell low enough, the next best seed word would be selected for processing. Formerly, all syntactic events had to be discarded before the next best seed word could be processed.

   Additional word matches were sought when "starting off" a seed event to guarantee a sufficient number of word-ending effects for anchored scans for adjacent words to proceed properly.

c) An increase in initial parallelism: If their scores are sufficiently close, up to three events from the top of the queue get handled in parallel.

d) The exclusion of certain words and classes (e.g., articles) from the initial scan, because their ubiquity in the grammar means their predictive power for adjacent words is poor.

e) A temporary removal of verification from the recognition strategy since its scores were incommensurate with those returned from lexical matching against the segment lattice. Verification scores are obtained and printed on the trace, so that statistics of the score distributions may be compiled [see Section I.E.].

These changes were implemented in versions of the HWIM system named "March 1", "March 15", and "March 31", after their approximate dates of creation. The system named "April 5" was almost identical to the March 31 system, with the single exception of updated APR statistics, which affect the lexical matching scores. Performance results for these

systems are given in the next section.

The behavior of the March 31 and April 5 systems, while less so than their predecessors, was still considerably depth-first. If presented with a "good" segment lattice, in which there were few serious APR errors and a correct word was among the top few seed events, chances were good that these systems would proceed fairly rapidly to a correct spanning theory. However, they were easily sidetracked onto unprofitable pathways from which their chances of trying other theories were slim.

The second phase of our control strategy development derived from a major change in scoring policy, based on a principle called shortfall density. This principle is more fully described in Section II.C. of this report, but briefly, it consists of scoring word matches and word match sequences not by their lexical-matching scores, but by the extent to which their lexical scores fall below a summed per-segment upper bound scoring function. This difference called the shortfall, is then divided by the duration, to give a shortfall density. The items on the event queue are ordered by increasing shortfall density, rather than by decreasing lexical score. It can be shown that following the partial hypothesis with the lowest shortfall or shortfall density guarantees the discovery of the best matching interpretation of the utterance.

A scoring policy operating behind such a guarantee is bound to be somewhat breadth-first, with the number of theories considered in the process of finding a spanning interpretation depending on the tightness of the upper-bound scoring function. Presently, we estimate this MAXSCORES function from the words returned during the initial scan and revise this estimate from subsequent anchored word matches if they score above it.

The implementation of the shortfall density algorithm was accompanied by four additional significant strategy changes.

a) Partial rectification of word scores. Formerly, the score for a sequence of adjacent, usually fuzzy word matches, was the sum of the scores of the best word match in each fuzzy. Now the score of the combination is the best sum of scores of word matches sharing common boundaries. This does not yet guarantee that the scored word matches are consistent at the boundaries, but is a step in that direction.

b) Elimination of the system of setting monitors for proposed words and remembering previously matched words in the word lattice. This concept was appropriate for noticing non-adjacent words based on semantic associations and for dealing with word matches in isolation, but is inappropriate for the syntactic island extension and anchored word matching of the present system. Words are now proposed and matched only in specific syntactic and lexical contexts.

c) An inclusion of the effect of "ghost words in scoring events." If A is the best word proposed and matched to the left of theory B, then any events extending theory B to the right also include the effect of A (the "ghost word") on the left, since no extension to the left can possibly do better than A's score.

d) A supplanting of shelving in favor of a strategy of blocking ghost words on the end of a theory opposite to the direction being extended. Let theory B propose and

match sets of words A and C on the left and right, respectively. That is, A is the left "ghost set" for any event Bc (where c C) and vice versa. If the extension Bc is processed, it may thereafter notice on its left only words not in its left ghost A. (Those words are reserved for the events aB (where a A).) Furthermore, its left ghost score is changed from the best score in the ghost set to the worst score, since any new words not in the ghost set must score at least that badly. This blocking and rescoring appears to offer the advantages of shelving without the damaging effects of removing events altogether.

These revisions to the control strategy did not result in a workable system until May 8. In order to be able to report development that took place during April, we are including the status and performance of the "May 8" system in this QTPR.

In terms of its other components, the May 8 system differs from the April 5 system in a slight change to the lexical matcher, a few additional across-word phonological rules, and an improved Syntax component. Verification scores are still decoupled from the strategy.

As shown by the success rates on 40 utterances given in the next section, the May 8 system cannot be demonstrated to be superior to its predecessor. The behavior of this strategy is indeed considerably more breadth-first, and many of its "failures" were due to hitting a time-out at 60 or 75 minutes of CPU time, with good events still on the queue. We have noted that it now tends to develop correct partial interpretations derived from different seeds, which then

complete with each other. A provision for "island collision," or merging compatible partial interpretations, should increase the effectiveness of the strategy in these cases.

2) System Performance

At the round of dress rehearsal demonstrations in February 1976, the SUR Steering Committee set as a task for each system-building site the selection and subsequent processing of 20 new utterances during each of March, April, and May. We had the SCRL group select the sentence types for these three sets, and we have been running our system primarily, but not exclusively, on them.

Although we digitized the 20 March utterances in late February, 10 of them became casualties of the move to another TENEX system in BBN's new office building. We therefore exercised our March systems on only 10 new utterances, all by speaker RMS. The results of the March systems on these 10 utterances are summarized below.

March 1	March 15	March 31
1/10	3/10	3/10

The texts of these sentences and the results themselves are presented below.

RMS110C	What is the registration fee?
RMS125	When is the next ASA meeting?
RMS137	How much is left?
RMS269	Please display all budget items.
RMS271	When is the next ACL meeting?
RMS273	Who went to Austin in November?
RMS274	The trip number is 5 4 3 8.
RMS275	Add a new budget item.
RMS277	The registration fee is twenty dollars.
RMS280	How many trips are there?

As it turned out, the voice characteristics of the speaker of those 10 March utterances were represented in our APR statistics (used for scoring lexical matches) by only one utterance. This led to poorer segment lattice scores than we were used to seeing. As explained in the previous section, our April 5 system used updated APR statistics that included these 10 RMS sentences, so for April and later systems, those sentences are no longer "new," but part (about 1/12) of the "design data". As expected, this resulted in a higher success rate of 6/7, as shown on p. 26. (Three of the 10 were not run with the April 5 system because of still-unfixed bugs.)

	March 1	March 15	March 31
RMS110C	** SUCCESS **	** SUCCESS **	** SUCCESS **
RMS125	Syntax ran out of space.	Failed to match ASA after NEXT.	Control ran out of space. Bad anyway, because ASA didn't match after NEXT.
RMS137	Got HOW MUCH IS, proposed but failed to match LEFT.	** SUCCESS **	Finished wrong: HOW MUCH -S IT TO FLY? (Different word match effects)
RMS269	Syntax ran out of space, was on a bad track anyway (too depth first).	Failed to match ITEMS after BUDGET.	Proposed but failed to match DISPLAY before ALL BUDGET ITEMS.
RMS271	Initial scan hopeless.	Got WHEN IS THE NEXT ACL, then proposed but failed to match MEETING.	Control ran out of space. Bad track anyway.
RMS273	Grammar bug: rejected ...TO BOSTON IN NOVEMBER.	(Same grammar bug.)	(Same grammar bug.)
RMS274	Syntax ran out of space, bad track anyway (too depth first).	Parser bug. Bad anyway: no match for IS after NUMBER.	Control ran out of space. Control bug anyway.
RMS275	Finished wrong: ENTER THE BUDGET ITEM.	Finished wrong: ENTER THE BUDGET ITEM.	** SUCCESS **
RMS277	Finished wrong: REGISTRATION FEES TO OUR NEW BUDGET. Grammar problem, at least.	Finished wrong: REGISTRATION FEES TO ITEM TWO. Grammar problem, at least.	Matched, but failed to accept DOLLARS after REGISTRATION FEE IS TWENTY. Parser bug.
RMS280	Syntax ran out of space. Proper final event was on the queue, though.	** SUCCESS **	** SUCCESS **

March results on March sentences.

In early April, the reassembly in our new building of the last TENEX system again made our real-time interface available. We were thus able to redigitize the "March sentences 11-20" as well as the 20 new April sentences. The April systems (including the one named "May 8") were exercised with these 40 utterances. The performance of the April 5 and May 8 systems is summarized below.

	March 31	April 5	May 8
March 1-10	3/10	6/7*	4/10*
March 11-20	--	3/10	3/10
April 1-20	--	2/20	2/20

*Utterances are included in APR statistics.

The results of the 30 new sentences are broken down into March and April sets because there appeared to be a significant performance difference. Possible reasons for such a difference are as follows:

a) Different recording conditions affecting the sentence acoustics. The April sentences were the first (and so far, only) ones recorded in our new building.

b) More difficult phonetic events. This seems to be one dimension of the way SCRL selected the sentences for our March, April, and May tests.

c) More difficult higher-level effects. This appears to be another SCRL selection factor. For example, the April sentences are longer.

We are currently investigating these effects, particularly the first.

(As another measure of performance, we ran the May 8 system on 10 "old" utterances that had previously been recognized by March systems, utterances with segment lattices known to be good. The new system got 9 of the 10 and timed out on the other one, with many correct partial interpretations on the queue.)

The job of exercising HWIM on a set of utterances has been facilitated by the implementation of a new program, RUNSYS. RUNSYS runs HWIM (as an inferior process) on each one in turn, producing trace files. The LISP processes of HWIM are set up so that if an error break occurs, they print out the contents of the stack and then terminate, so as to return control to RUNSYS. They also terminate gracefully if the CPU time used exceeds a threshold. RUNSYS monitors the system load and defers running HWIM if the load is too heavy, so as not to interfere with interactive users. Should the TENEX system crash, RUNSYS is restarted when the system comes back up.

The following pages give the texts for the 30 new utterances and the results of the April 5 and May 8 systems on all 40 utterances.

The 30 New Sentences

JJW102B What is the total budget figure?

JJW119B Who's going to IFIP?

JJW126B	How much have we already spent?
JJW129	When is the IFIP conference?
JJW173	What is the plane fare there?
JJW270	When did Craig go to Utah?
JJW272	What is the plane fare to Ottawa?
JJW276	Create a budget item.
JJW278	List the remaining untaken trips.
JJW279	Which trips were canceled?
JJW281	When is Bill going to Washington?
JJW292	Schedule a trip by train to New York.
JJW294	What are the final budget figures?
JJW296	List the final costs for those trips.
WAW282	Schedule a trip for Bonnie to Washington.
WAW283	Cancel Lyn's trip to the ASA meeting.
WAW284	What do we have budgeted for the IFIP conference?
WAW285	Give me a list of the untaken trips.
WAW286	What trips have been taken this year?
WAW287	List all trips to New York this quarter.
WAW297	Am I going anywhere in November?
DHK113	Is John scheduled to go to Carnegie?
DHK288	How much did we spend in 1975?
DHK289	Change the number of people from three to two.
DHK290	What trips remain in the robot budget?
DHK291	How much money is in the current budget?
DHK293	How long will the IFIP meeting be?
DHK295	Do you have any trips in July?

DHK298 Enter a trip for Jerry to San Francisco.

DHK299 Where is the ASA meeting this year?

10 March sentences, using APR statistics including them

	April 5	May 8
RMS110C	** SUCCESS ** (7 theories, 919 sec.)	** SUCCESS ** (20 theories, 1038 sec.)
RMS125	Failed to match ASA after NEXT, so stopped.	Got instead: WHEN IS THE NEXT D.C. MEETING. Poor match for ASA. (31 theories, 1674 sec.)
RMS137	** SUCCESS ** (3 theories, 364 sec.)	** SUCCESS ** (20 theories, 1825 sec.)
RMS269	** SUCCESS ** (10 theories, 1348 sec.)	Failed to match PLEASE before DISPLAY ALL BUDGET, due to lexical retrieval bug.
RMS271	** SUCCESS ** (25 theories, 1143 sec.)	Timed out: 75 min. Had got WHEN IS THE NEXT.
RMS273	(not run)	Syntax ran out of PCONFIG space.
RMS274	(not run)	Got instead: THE TRIP NUMBER IS 54488. (31 theories, 2890 sec.)
RMS275	** SUCCESS ** (9 theories, 825 sec.)	** SUCCESS ** (28 theories, 1777 sec.)
RMS277	(not run)	Timed out: 60 min. THIRTY matched slightly better than TWENTY, so at best it could only have got it with "$30".
RMS280	** SUCCESS ** (9 theories, 787 sec.)	** SUCCESS ** (70 theories, 3618 sec.)

The 30 new sentences

	April 5	May 8
JJW102B	** SUCCESS ** (10 theories, 1070 sec.)	Got instead: WHAT'S BATES'S TOTAL BUDGET FIGURE
JJW119B	Failed to match GOING before TO IFIP.	IFIP is the only correct seed, and it never gets worked on. Timed out.
JJW126B	** SUCCESS ** (27 theories, 1119 sec.)	** SUCCESS ** (4 theories, 2778 sec.)
JJW129	** SUCCESS ** (21 theories, ? sec.)	Hit error break in Syntax.
JJW173	Hit grammar bug.	Failed to find PLANE-FARE before THERE.
JJW270	** SUCCESS ** (38 theories, 1587 sec.)	** SUCCESS ** (79 theories, 3521 sec.)
JJW272	Best event TO OTTAWA fails to get worked on.	** SUCCESS ** (36 theories, 2694 sec.)
JJW276	Failed to find ITEM after BUDGET.	Timed out just as A BUDGET reached top of the queue.
JJW278	Got instead: LIST REMAIN-ING TAKEN TRIPS. (!)	Got instead: LIST TWO REMAINING UNTAKEN TRIPS.
JJW279	No correct seed events.	** SUCCESS ** (Note that it got WHAT TRIPS WERE CANCELED) (46 theories, 3700 sec.)
JJW281	Run stopped, perhaps prematurely.	WHEN IS never gets worked on; failed to find GOING before TO WASHINGTON.
JJW292	Failed to find NEW YORK after TRIP BY TRAIN TO.	Fails to find BY after TRIP.
JJW294	Very poor match for BUDGET before FIGURES, only good seed, so stopped.	Good events from the 3 correct seeds get pushed way down the queue, don't get worked on.
JJW296	Doesn't get to work on any good seeds. Stopped.	Gets FOR THOSE TRIPS and finds COSTS, but that event doesn't get to the top of the queue before timeout.

WAW282	No correct seed words.	** SUCCESS ** (28 theories, 4018 sec.)
WAW283	Doesn't get to work on any good seeds. Stopped.	Gets -S TRIP TO THE ASA MEETING and finds LYN, but that event never gets worked on before timeout.
WAW284	Failed to find BUDGETED before FOR THE IFIP CONFERENCE.	Failed to find BUDGETED after WHAT DO WE HAVE.
WAW285	No correct seed words.	Bad match for UNTAKEN before TRIPS, other seed LIST never seen.
WAW286	Failed to find TRIPS before HAVE BEEN TAKEN THIS YEAR.	Failed to find TRIPS before HAVE BEEN TAKEN THIS YEAR.
WAW287	Syntax rejected NEW YORK before THIS QUARTER.	Shortfall scoring problem causes good events to be killed.
WAW297	Only good event IN NOVEMBER pushed way down the queue. Stopped.	Events AM I and IN NOVEMBER pushed way down the queue.
DHK113	No correct seed words.	Only good seed SCHEDULED never gets worked on. Poor segment lattice.
DHK288	Only good event pushed way down queue. Stopped.	Failed to find NINETEEN after HOW MUCH DID WE SPEND IN.
DHK289	Only good event pushed way down queue. Stopped.	Good events from only seed (PEOPLE) fall down queue, don't get worked on.
DHK290	** SUCCESS ** (9 theories, 808 sec.)	Notices ROBOT after WHAT TRIPS REMAIN IN THE, but shortfall scoring quirk kills it, even though it's a good match.
DHK291	Failed to find MONEY after HOW MUCH.	Gets as far as MONEY -S IN THE CURRENT and -S IN THE CURRENT BUDGET, but problems with poor match for MONEY and a possible control bug.
DHK293	Control strategy bugs.	Ran out of space in parser.

DHK295	Failed to match IN before JULY.	Got instead: DO YOU HAVE THE TRIPS. Failed to find ANY before TRIPS.
DHK298	No correct seeds.	Best seed starts as #9 on queue, never gets worked on.
DHK299	Best seed starts as #3 on queue, never gets worked on.	Single good seed (MEETING) gets poor matches for ASA and THIS on either side, so they go way down the queue.

References

Bahl, L.R., J.K. Baker, P.S. Cohen, N.R. Dixon, F. Jelinek, R.L. Mercer and M.F. Silverman (1976)
"Preliminary Results on the Performance of a System for the Automatic Recognition of Continuous Speech," Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, Philadelphia, April 12-14, pp. 425-429.

Bruce, B. and G. Harris (1974)
"Procedural Semantics in the Travel System," Speech Understanding Systems, Quarterly Technical Progress Report No. 3, Report No. 3115, Bolt Beranek and Newman Inc. Cambridge, Mass., pp. 27-36.

Earley, J. (1970)
"An Efficient Context-Free Parsing Algorithm," CACM, 13:2, February, pp. 94-102.

Schwartz, R. (1976)
"Acoustic-Phonetic Experiment Facility for the Study of Continuous Speech," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, April 12-14, Philadelphia, pp.1-4. (Also in Woods et al., Speech Understanding Systems, Quarterly Technical Progress Report No. 5, Report No. 3240, Bolt Beranek and Newman Inc., Cambridge, Mass., pp. 62-73.

Schwartz, R. and V. Zue (1976)
"Acoustic-Phonetic Recognition in BBN SPEECHLIS," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, April 12-14, Philadelphia, pp. 21-24. (Also in Woods et al., Speech Understanding Systems, Quarterly Technical Progress Report No. 5, pp. 47-61

Woods, W.A. M. Bates, G. Brown, B. Bruce, C. Cook, L. Gould, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue (1975)
"Speech Understanding Systems, Quarterly Technical Progress Report, No. 4," Report No. 3188, Bolt Beranek and Newman Inc., Cambridge, Mass.

II. TECHNICAL NOTES

A. Evolving Uses of Knowledge in a Speech Understanding System

Bonnie Nash-Webber
Bertram Bruce

1) Introduction

1.a. The BBN Speech Understanding System

1976 is the fifth year in a five-year program on automatic speech understanding which is being carried on, under the sponsorship of ARPA, here at BBN and at various other sites across the country. Over these years, the character of our speech understanding system, hereafter called HWIM ("Hear what I mean"), has undergone substantial changes. Many of these changes involved either the introduction of new components, the restructuring of existing ones, alterations in information flow between components, or altered use of that information. The purpose of this paper is twofold: to describe some of the changes that have happened to our system, and to present some common sense principles of system dynamics that might be abstracted from them.

In the following discussion, we will be using the term "knowledge source" or KS to refer to a process or a set of processes and their attendant data structures, whose job it

is to provide the system and/or other KS's with a particular kind of information (knowledge). A KS is a conceptual entity, one of the building blocks of our conceptual view of the system. "Component", on the other hand, will be used to refer to the embodiment of a KS or part of a KS in the actual system. For example, our phonological knowledge source is embodied partly in our Dictionary Expansion component and partly in our Lexical Retrieval component.

1.b. Knowledge Sources in HWIM

There are several types of knowledge being accessed in the HWIM system: acoustic-phonetic, phonological, lexical, syntactic, semantic, discourse level, and factual. A brief description of each of these is given below.

Acoustic-phonetic knowledge can be characterized in two directions. Analytically, it is of the relationship between observable acoustic phenomena and the possible phonemes or phoneme sequences underlying them. In a generative mode, it includes knowledge of an "ideal" parametric representation of a string of phonemes and ways in which actual utterances of that string can vary from that ideal.

Phonological knowledge includes knowledge of valid and/or likely phoneme sequences within words of the language and the way in which the realizations of phonemes may change in the context of other phonemes, both within words and

across word boundaries.

Lexical knowledge includes the possible base pronunciations of each word in the lexicon, both stylistic variations - /aft n/ versus /ɔfən/ - and functional variations - /pra´ jɛkt/ as a noun, versus /prə jɛkt´/ as a verb. The knowledge of how regular nouns, verbs, adjectives and adverbs inflect is also subsumed under lexical knowledge.

Syntactic knowledge is knowledge of what word sequences can be produced from a given vocabulary if questions of meaning are ignored. This knowledge, embodied in a grammar, makes it possible to derive the grammatically acceptable contexts for any given word, phrase, or acceptable word string, judgments about the potential grammaticality of some other word sequence, and the locations of phrase boundaries that correlate with speaking patterns.

We are taking semantics to mean "the scientific study of the relations between signs or symbols and what they denote or mean." As such, semantic knowledge includes, among other things, knowledge about the meaningfulness of a word sequence, its completeness with respect to denotation or, conversely, its oddity, and also knowledge of the contexts in which any word, phrase, or acceptable sequence of words can meaningfully occur. It does not include, under the above definition, knowledge of the situation in which the

utterance is made, nor of the presuppositions underlying utterances: these we have characterized as being discourse-level knowledge.

Discourse-level knowledge, based on both idealized dialogue models and the on-going discourse and situation it establishes, includes knowledge of appropriate and/or expected types of replies to given types of questions and also of the current set of objects and events available for reference at each stage of discourse.

Factual knowledge is domain- and task-specific knowledge. In HWIM's current travel budget management domain, it includes the trips that have been taken or planned, the travel budgets for various projects and how parts of those budgets have been allocated for various purposes, the fares for traveling between cities, and all the specific people, cities, institutions, etc., that might be or have been involved in a trip.

An important point to remember is that there is not, a priori, any single best way to use these sources of knowledge in a speech understanding system. A system that relies too heavily upon a fixed, limited conception of how KS's should interact is apt to founder when in contact with real speech input. Similarly, one that envisions each KS interacting with every other KS, producing all the information that might theoretically be of use, is likely to

die of uncontrolled activity. Design decisions for the HWIM system at BBN have been based on theoretical principles, but they have been refined in response to specific problems encountered in understanding speech. At any stage in our system's evolution, the way in which a given KS is accessed - when, by whom, and to what end - has depended on the capabilities of the other KS's in the system and reflects the maximum leverage the given KS can provide the entire system in accomplishing its goal.

1.c. HWIM's Task Domain

HWIM is designed to serve as a travel budget manager's automated assistant, helping the manager by keeping a record of upcoming conferences, trips taken or proposed, and money allocated for future trips. It also serves as a source of information about people, places, projects, contracts, and other data relevant to budget maintenance. The particular task of travel budget management is a simple example of the general resource management problem and is an initial step towards an intelligent manager's assistant.

Utterances processed by HWIM are assumed to be appropriate to the travel budget management domain, and more specifically, of the type which the manager (rather than the assistant) would utter. Thus, "What is the air fare to Chicago?" would be a typical utterance, whereas "You are

over your budget!" would not. A hypothetical dialogue which illustrates some characteristics of HWIM's task domain is shown in Figure 1. (See also Figure 3.)

- - - - - - - - - - - - - - - - - - - -

Manager: How much is left in the budget?
HWIM: There is $743.18 remaining in the speech understanding budget.

Manager: Estimate the cost of a trip to St. Louis for one person for three days.
HWIM: The estimated cost would be $312.57.

Manager: When is the next ASA meeting?
HWIM: November 13-17, 1976.

Manager: Where is it?
HWIM: Chicago.

Manager: What is the fare from St. Louis to Chicago?
HWIM: The one-way air fare is $78.

Manager: Schedule a trip for Chip to St. Louis and Chicago.
HWIM: When is he leaving Boston?

Fig. 1. A hypothetical dialogue between a travel budget manager and HWIM.

2) The First Incarnation of HWIM

2.a. HWIM as SPEECHLIS

When HWIM (then called SPEECHLIS) first emerged in 1973 as a complete speech understanding system, it comprised an acoustic-phonetic knowledge source (KS), a phonological KS, a lexical KS, a syntactic KS, a semantic KS and above them, a control program directing their activities. It did not have a source of either discourse-level or generative acoustic-phonetic knowledge, and while its data base and retrieval system did together constitute a source of factual knowledge, no effort had been made to integrate it into the understanding process. The specific task undertaken in this first version of HWIM was to answer questions a geologist might pose about the Apollo 11 lunar rocks, specifically their composition and age analyses. For example,

> Give me olivine analyses for the breccias.
> Does sample 10004 contain plagioclase?
> What is the age of each fine-grained igneous rock?

Figure 2 illustrates the structure of this first version of HWIM. Each KS was embodied in one or more components of the system. In the figure, components are represented as labeled boxes and control and communications paths as lines between them. (In the following discussion, components and significant data structure are underlined when first referenced.)

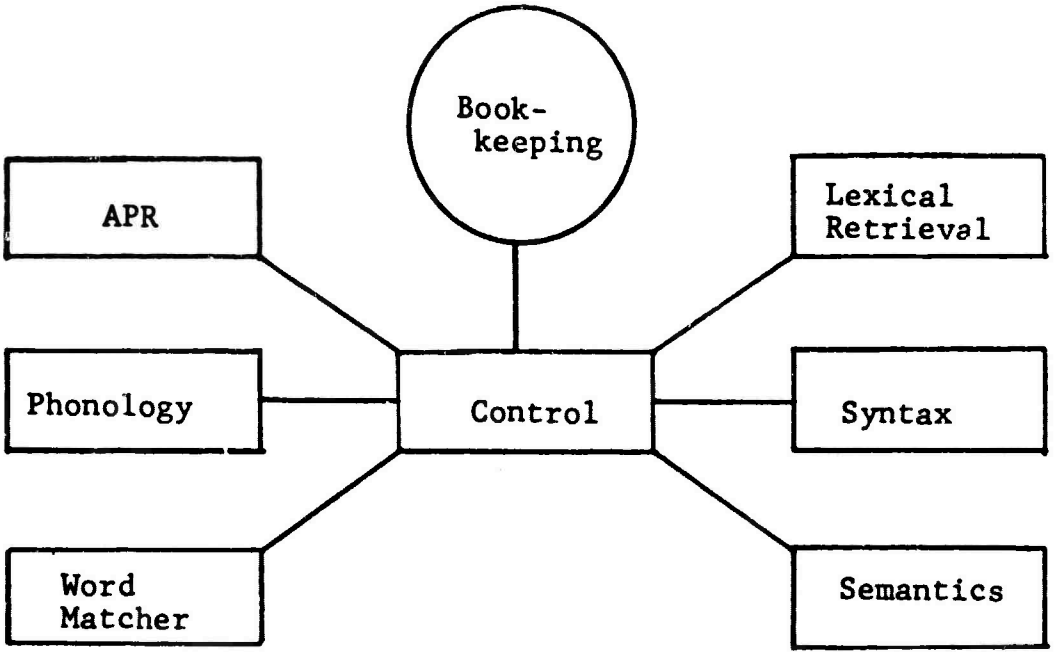


Figure 2.

2.b. How Knowledge was Used in the Original HWIM

The ways in which this first version of HWIM used its knowledge sources were the following. The acoustic-phonetic KS (Acoustic-phonetic recognition or APR component) performed a first-pass segmentation and labeling of the acoustic signal into partial phonetic descriptions, producing as output a segment lattice that compactly represented all possible alternative segmentations of the utterance and the alternative identities of the individual segments. Phonological knowledge was represented in the form of analytic phonological rules. After the APR

component produced its initial segment lattice, the phonological component augmented the lattice with branches representing possible underlying sequences of phonemes which could have resulted in the observed acoustic sequences. Associated with each added branch was a predicate function that could later be used by the Word Matcher component to check for the applicability of the given phonological rule based on the specific word spelling and necessary context.

The Lexical Retrieval component, embodying part of HWIM's lexical knowledge source, was used to retrieve words from the lexicon to be matched against the input signal. The Word Matcher component, embodying the other part, when given a particular word and a particular location in the input signal, would determine the degree to which the word matched the segment lattice. Word matches that scored above a certain minimum would then be available for manipulation by the rest of the system.

HWIM's Syntactic KS, realized in its Syntactic component, was used to judge the grammaticality of a hypothesized (possibly partial) interpretation of the utterance and to make limited proposals of words and/or syntactic categories to extend a partial interpretation. These partial interpretations were represented in data structures called theories, which contained one or more non-overlapping word matches, as well as information

describing their possible syntactic and semantic associations. The Semantic component was used to notice coincidences between semantically related words that had been found at different places in the signal, to judge the meaningfulness of a hypothesized interpretation, and to predict particular words or specific semantic classes of words for extending a partial interpretation.

2.c. Performance Characteristics of the Original HWIM

The above first characterization of the necessary knowledge sources and how they might potentially contribute to the overall speech understanding task was based on experience gained through "incremental simulations" of a complete speech understanding system [Woods and Makhoul, 1974]. This initial characterization was then modified by the then-current and expected future performance of the actual components (i.e., their speed in providing a specific type of information and the reliability of the information provided). A description of the relevant performance characteristics of each component will perhaps explain its particular (and limited) use in the 1973 system.

At that time, the speed of the Lexical Retrieval component was strongly dependent on the size of the dictionary and less so on the complexity of the segment lattice. Thus, phonological knowledge, which had to be used

somewhere to account for contextual variations in pronunciations, was applied analytically to the segment lattice, instead of generatively to the lexicon. This latter method would have at least tripled the size of the dictionary. To account for the inflected forms of regularly inflected words like "contain" ("contains", "contained", "containing"), inflectional endings were matched against the segment lattice only after a match for the base form had already been found. The alternate method of including all regularly inflected word forms in the dictionary would have expanded it by about 40% and again possibly crippled the Lexical Retrieval component.

The reliability of the Word Matcher component depended, of course, on the ability of the APR component to produce as narrow and accurate a segmental characterization of the utterance as possible. At this point in HWIM's development, this characterization was relatively broad, resulting in a large potential for spurious, incorrect word matches. The longer a word, however, the less chance a good match of it was spurious. As a result, only proposals for long words were encouraged. In addition, the Word Matcher had no efficient way to match several words simultaneously: the time required to match N words was N times the time required for one. Hence, only very limited predictions were made by other components. In order for processing to begin, a single data-driven context-free scan was done for the best

matching long words, in cooperation with the Lexical Retrieval component. Though expensive, it could be relied upon to return several correct word matches among a not inordinate number of spurious ones.

Based on its general broad grammar for English, the Syntactic component could neither judge the potential grammaticality of a short sequence of word matches nor make limited predictions of the context in which the sequence could occur. Thus it could not be used effectively on short theories. For long theories though, Syntax could be an effective judge of grammaticality. However, with respect to predicting the possible contexts of even long theories, predictions could only be made in terms of short function words like "the" or "in," or large general categories like adjective or verb. The latter posed too much of a burden on the Word Matcher in terms of time, and the former, unless they were the only possibilities, did not yield reliable matches. Thus the Syntactic component could only be used effectively for evaluating long theories and making predictions to the Word Matcher when the set of possibilities was extremely constrained.

In HWIM's lunar rock domain, the range of semantic relationships possible among the entities was relatively small. Thus the Semantic component could be used to decide which of the word matches found on the initial scan could

meaningfully go together in a single utterance, without the number of such possible combinations being unmanageably large. With respect to proposals, because names and descriptions of the entities in the domain tended to involve long words (e.g., "fayalitic olivine", "fine-grained igneous rocks", etc.), one could be fairly confident of the results of Semantics' predictions. However, except with respect to names and descriptions, Semantics had no information about possible word order that would enable it to make proposals. Coupled with the small number of "content" words that might reasonably be expected in an utterance - usually no more than half the words, the others being syntactic function words - the primary uses of the semantic knowledge were just those given above.

3) Evolution of the HWIM System

3.a. Addition of New Knowledge Sources

One type of change that has occurred in the evolution of HWIM has been the addition of new sources of knowledge. Since its emergence in 1973, three new knowledge sources have been incorporated into the system: a discourse-level KS, a factual KS, and a generative-acoustic KS. The addition of a new knowledge source raises three basic questions: how, when, and where to use the information the KS can provide. For example, a discourse-level KS could

potentially provide such information as the objects and events available for anaphoric reference, the likely gist of the current utterance based on the state of the discourse, and likely (or unlikely) ways the utterance might be phrased. This information could be used in several ways:

(1) Predictively, when making a first pass over the utterance looking for good word matches to anchor on. E.g., "I expect the manager to want to fix the data base; therefore look for 'edit' verbs and words describing a data base structure, like a trip or meeting descriptor."

(2) Evaluatively, when assigning a score to a theory. E.g,. "Lower your confidence in this theory containing "their", as I have no referent for it."

(3) To relax constraints, say on grammaticality, when a theory is being evaluated by Syntax. E.g., "A single noun phrase denoting a city is acceptable as a complete utterance here."

(4) Comparatively, when deciding what hypothesis to pursue further. E.g., "The theory for "their fare" is preferable to that for "the air fare" since I believe a train trip is under discussion."

Theoretically, any or all of the above uses would be valuable. However several factors argued that the greatest immediate leverage for the least effort could be obtained from using discourse-level information in the now-combined Syntactic-Semantic knowledge source (see Section C.3) to relax constraints on grammaticality. Other types of discourse-level information that were not going to be used were then not produced, nor were other KS's that might have used the information modified to do so. This illustrates

one of the common sense principles we wish to get across:

> Principle: A KS only provides information to those KS's which can use the information efficiently to improve overall system performance.
>
> Corollary: If there is no KS which can efficiently use some type of information potentially available from a given KS, the information is not provided and may not even be computed.

To return to the factors that argued for using discourse-level information to relax constraints on grammaticality, we list three below.

(1) One mode in which HWIM operates is that of a directed dialogue, with the system asking questions and the user replying to them. A sample of such a dialogue is given in Figure 3. Notice three things: the system expects a valid answer to its question, the meaning of the answer can be characterized as belonging to a particular semantic class (e.g., date, amount, location, etc.), and the form of the answer is rarely a complete sentence. Now these types of incomplete, but reasonable answers map directly onto the kinds of constituents parsed by the pragmatic grammar. To permit such answers, we need merely take the current discourse-level expectation (noted in brackets in Figure 3), expressed simply as DATE, AMOUNT, LOCATION, etc., and allow the Syntactic-Semantic component to accept the corresponding constituent as a possible complete utterance. (To open the grammar and freely allow stand-alone noun phrases to be

accepted anywhere in the discourse would lead to lower system performance, since outside this directed dialogue, we expect utterances to be complete sentences.)

(2) The current reliability and speed of the now-combined Lexical Retrieval/Word Matcher component (hereafter referred to simply as "Lexical Retrieval") is such that the context-free initial scan of the segment lattice for the 15 best matching words takes approximately one CPU minute for a 2.5 second utterance and most often turns up at least 2 correct words among the top 15. One could not buy much by constraining the set of words sought on this scan to only those expected. In addition, there would remain the inevitable problem with all constraints that the correct answer may be improperly constrained out.

(3) In order to implement a procedure for comparing theories on the basis of discourse-level knowledge, we would have needed much more global information concerning the existing set of theories and their inter-relationships than we currently keep around. This would have put a large burden on limited space resources. More importantly, it would have required valid criteria for making such relative evaluations, and we currently do not have them, either theoretically motivated or based on our experiences.

A DIALOGUE WITH THE TRAVEL BUDGET MANAGER'S ASSISTANT

Manager:	What is the air fare from Boston to Baghdad?	
HWIM:	THE ONE-WAY BOSTON TO BAGHDAD AIR FARE IS $643.	
Manager:	Enter a trip for John Makhoul to Baghdad.	
HWIM:	WHEN IS THE TRAVELER LEAVING BOSTON?	[DATE]
Manager:	November twenty-fifth, nineteen seventy five.	
HWIM:	WHEN IS THE TRAVELER LEAVING BEIRUT?	[DATE]
Manager:	December eleventh.	
HWIM:	WHAT IS THE COST FOR THIS TRIP?	[AMOUNT]
Manager:	Fifteen hundred and fifty dollars.	
HWIM:	WHAT IS THE TRIP NUMBER?	[DIGIT STRING]
Manager:	Eight eight seven three.	
HWIM:	TO WHAT CONTRACT IS THIS TRIP TO BE CHARGED?	[CONTRACT]
Manger:	Speech compression.	
HWIM:	THE FOLLOWING TRIP HAS BEEN ENTERED: JOHN MAKHOUL TO BEIRUT, LEAVING BOSTON ON 25-NOV-75 AND BEIRUT ON 11-DEC-75. THE TRIP NUMBER IS 8873 AND THE TRIP IS CHARGED TO THE SPEECH COMPRESSION CONTRACT.	
Manager:	Okay.	

Figure 3.

Though not yet implemented, we do envisage the Syntactic-Semantic component using other discourse-level information, that having to do with referents, in its evaluation of a theory. That is, from its pragmatic grammar it knows what kind of entity it is assuming a given pronoun references. If there is no referent for an entity of that kind in the discourse, it can reduce its confidence in the theory at a very small cost.

3.b. Changes in Existing Knowledge Sources

Another set of changes that have occurred in HWIM since its first appearance in 1973 have involved the way in which knowledge is represented in a knowledge source. These changes have resulted in more efficient and reliable use of that knowledge. Two such examples are discussed below.

In the first stages of HWIM's development, as mentioned earlier, the word matcher component, which embodied most of the system's lexical knowledge, was relatively slow in performing the task of matching a word against a region of the utterance. It also had no mechanisms which would reduce the amount of time needed to match N words en masse, below that of matching N words individually. Thus while it might have been possible to get from another KS an exhaustive enumeration of the words acceptable adjacent to a given word match, it would not have improved overall system performance at that time.

In early 1975, HWIM's lexical knowledge source was re-implemented in the form of an improved Lexical Retrieval/Word Matcher component. Not only was it much faster at matching individual pronunciations against the segment lattice, but it was also able to handle multiple matching very efficiently, based on a distributed key representation of the lexicon in which common parts of pronunciations are merged [see Section II.B.]. (For example, it currently takes on the average only 6.83 CPU seconds to effectively compare 448 words against a given position in the segment lattice.) In addition, and this will be more relevant to another change in the system to be discussed later, it was not strongly affected by dictionary size. As a result of this change in Lexical Retrieval, the Syntactic component was modified to produce an exhaustive enumeration of all words and syntactic categories that could grammatically occur adjacent to a given word match or sequence of word matches. This re-illustrates our first principle of system dynamics: information, though potentially available, should not be provided (and may not even be computed), unless or until some KS can use that information to improve overall system performance.

A second one of HWIM's existing knowledge sources whose organization and use has changed over time is its phonological KS. Initially phonological knowledge was represented in the form of analytic phonological rules that

were applied to the basic segment lattice to account for possible underlying sequences of phonemes which could have produced the observed acoustics. Though this use was forced on us by the inability of the Lexical Retrieval component to handle a large dictionary, it was unsatisfying on several grounds.

First, it required applying phonological rules to data that was not certain (i.e., the segment lattice) rather than to data that was (i.e., the pronunciations given in the lexicon).

Secondly, it was difficult to represent deletion transformations in the segment lattice.

Thirdly, the rate of increase in the complexity of the segment lattice as new analytic phonological rules were added was such that we foresaw that the segment lattice would become unmanageable long before we completed the implementation of the thirty analytic rules envisioned. (With only eleven of the thirty rules implemented, there was a three-fold increase in the number of acceptable word matches being returned from the initial scan. While more correct words were also being returned, improving system performance for the time, it was predicted that as more rules were added, the correct matches would be swamped by the spurious ones.)

The ability of the new Lexical Retrieval component to handle large dictionaries efficiently meant the phonological knowledge source could be reorganized around generative phonological rules applied directly to the dictionary, rather than analytic rules applied to the segment lattice. This reorganization resulted in a Dictionary Expansion component as the embodiment of the phonological KS. This component also contained lexical knowledge about regular inflections, since all inflected forms of regular words could now be compiled into the dictionary as well, thereby eliminating the inefficient and frequently damaging recourse of only matching inflectional endings subsequent to a match for the base form.

(Figure 4 shows the expansion of our current dictionary of 507 base forms - TRAVELDICT - through compiling in inflection information and various types of generative phonological rules. Corresponding numbers are also given for a larger dictionary - BIGDICT - we plan to adopt in the near future. The phases of expansion reflect: (1) producing regularly inflected word forms, (2) applying ideal phonological rules to the one or more base forms of each word, (3) applying phonological rules that reflect either allophonic variations or APR dependency, and (4) accounting for potential across-word boundary phonological effects.)

The above change in the phonological knowledge source illustrates another of our common sense principles of system dynamics:

> Principle: The form in which a KS packages the information it is asked to provide and the circumstances under which that information will be used depend upon characteristics of the recipient KS.

	TRAVELDICT	BIGDICT
Roots & irregularly inflected forms	507	1072
↓ phase 1		
Roots & all inflected forms	702	1437
Pronunciation baseforms	818	1619
↓ phase 2		
Pronunciations following 1st application of phonological rules	1562	3484
↓ phase 3		
Pronunciations following 2nd application of phonological rules	1910	4288
↓ phase 4		
Pronunciations following application of across-word phonological rules	3940	7047

Figure 4.

3.c. Combining of Existing Knowledge Sources

A third important type of change that has occurred in HWIM has been the fusing of existing knowledge sources in order to improve overall system performance. The improvement comes from being able to constrain as soon and as tightly as possible the acceptable ways in which a given theory can be extended.

In the first version of HWIM, the syntactic and semantic knowledge sources operated independently in determining allowable extensions and making appropriate word predictions. Obviously, in many cases words acceptable syntactically would not be acceptable semantically and vice versa. Cooperation appeared to be needed at the individual word level. For example, an analysis of the travel budget management domain showed that following the words, "Are we over ... " only a small set of phrases are both syntactically and semantically plausible. Whereas Syntax would allow the past participle of a verb, say "taxed", or a noun phrase, say "the hill", the addition of semantic and pragmatic constraints of the travel budget domain restricted the plausible phrases to such things as "our budget this quarter" or "-budgeted".

To effect the close cooperation between Syntax and Semantics that was needed to make precise word predictions, a new KS was formed that merged the original syntax KS with much of the knowledge originally built into the semantics KS. (The semantic knowledge relating parse trees and their formal "meaning" to the system, as well as the programs for effecting this translation, have not been restructured and incorporated into this new KS, mainly for expedience. Future work on HWIM may see this fusion completed, if it leads to improved system performance.)

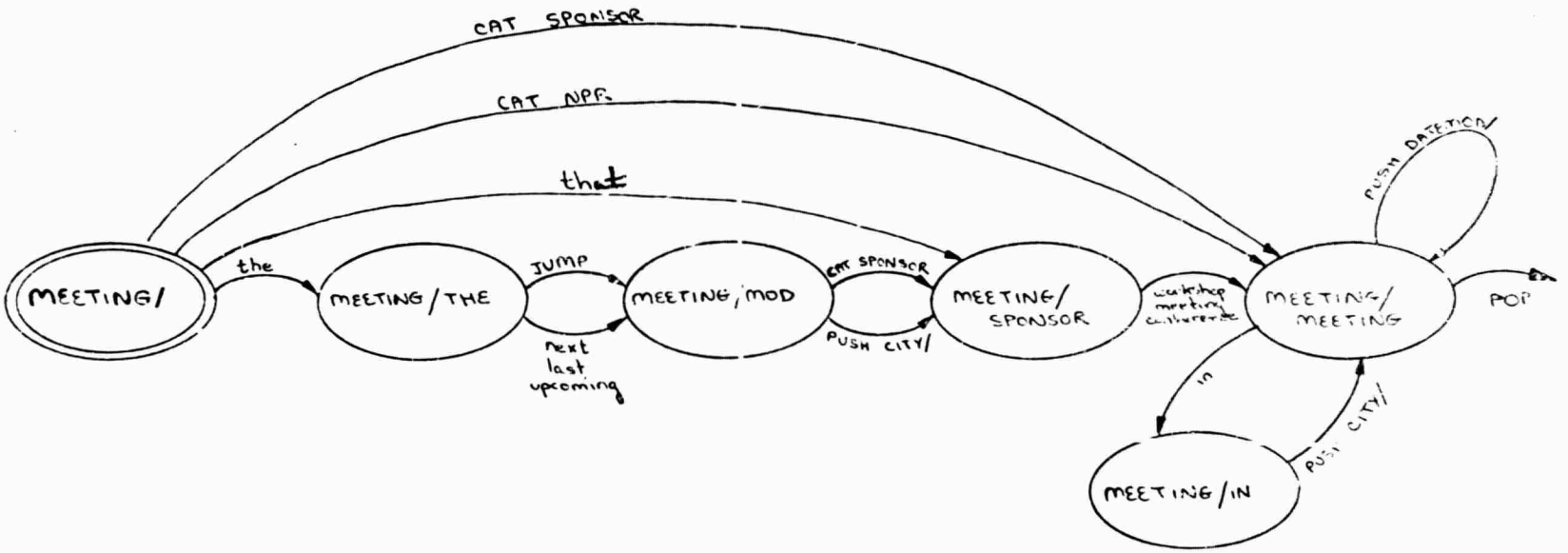
The resulting KS is built around a "pragmatic grammar." This grammar is an information structure in the augmented transition network (ATN) formalism [Woods, 1970]. It provides tremendous predictive power by accepting only those utterances that are grammatical in the usual sense, meaningful in the travel budget management domain, and appropriate to the pragmatic circumstance of a single speaker talking to a computer data management system. For example, while it allows "Enter a trip for Bonnie to Ottawa," it does not allow "Are you going to Ottawa?" which presupposes a traveling computer.

The pragmatic grammar does not have the usual PP, NP, and VP constituents. Instead, its non-terminal elements are structures such as "meetings", "trips", and "expenses". Figure 5 shows a fragment of the grammar, which parses

meeting and conference descriptions. It accepts (generates) a variety of reasonable meeting descriptions, but not phrases such as "her workshop" or "the air fare meeting", which, although constructed of words in the HWIM vocabulary, acceptable in a normal English grammar, and meaningful in certain contexts, have been determined to be outside the domain of a travel budget manager talking to his automated assistant.

The fusion of Syntax and Semantics into a single KS has been a major factor in HWIM's much improved performance, aided by a faster and more efficient parser. It demonstrates to us that a naive conception of KS interaction, which assumes that if communication channels exist, they will be used effectively, is wrong, at least in terms of currently realizable systems of HWIM's size and complexity. It suggests another obvious, yet often ignored, principle:

> Principle: If it is found that one must frequently consider simultaneously information from several KS's, then the activity of those KS's should be tightly coupled.



Examples: "the next ACL meeting"
"the Pittsburgh workshop in April"
"Carnegie for a week"
"that conference in Ottawa"
"IJCAI"

Fig. 5. A fragment of the pragmatic grammar used by HWIM.

4) Current Status of the HWIM System

Having focused in this paper on some of the changes that have occurred in HWIM over these past three years, it might be useful to present a brief overview of HWIM's current state.

At present, HWIM is operating on a dictionary of 702 words (507 roots), of which 419 are known to the pragmatic grammar. Work is still in progress towards extending the grammar to cover the full set of 702 words, as well as the larger dictionary of 1337 words mentioned earlier. Both of these extensions require enlarging the conceptual range of the grammar: the new areas within the travel budget management domain that become open for discussion will depend upon what information is, or might reasonably be, available from the factual data base and its ease of access. (This is yet another example of the system inter-dependencies that shape its ultimate characterization.)

We shall be giving this brief overview of HWIM's current state in terms of its components, rather than its knowledge sources, as we feel that this is a clearer way to show how and when each KS is being used. A somewhat fanciful schematic of the current version of HWIM is given in Figure 6, with ovals indicating components; dashed lines, the communication channels between them; amorphous blobs,

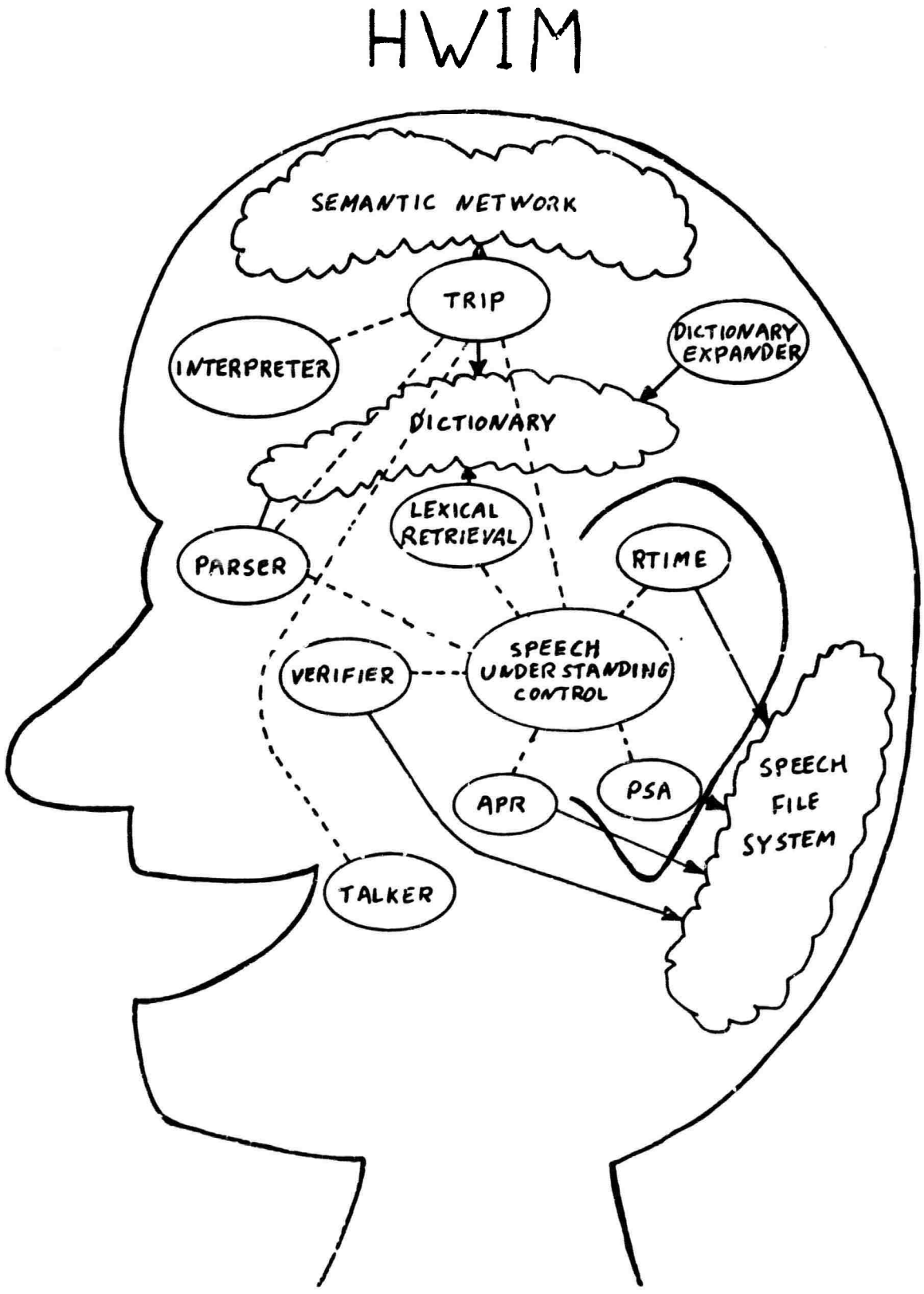


Fig. 6. The conceptual structure of the HWIM speech understanding system.

data structures accessed by more than one component; and arrows, access or manipulation of a data structure by a component. (In the following presentation, the names of components will be underlined when first referenced.)

The Dictionary Expansion component is used before any processing of individual utterances begins to expand the dictionary to encompass all predictable pronunciation variants. Though this is a time-consuming operation, it need only be done once on a given dictionary [Woods and Zue, 1976].

RTIME is HWIM's real-time interface and is used to acquire and digitize a new utterance, which is then parameterized by the component labelled PSA. These two components have been part of HWIM all along, but, as they have not been thought of as knowledge sources, we have omitted them from discussion in this paper.

The APR component takes the parametric representation of the utterance and applies acoustic-phonetic knowledge in a bottom-up (analytic) manner to produce a segment lattice representation [Schwartz and Zue, 1976].

The Lexical Retrieval component, which embodies the whole of HWIM's lexical knowledge source, is used in two ways in the understanding process: (1) to make an initial context-free pass over the segment lattice, looking for the

n best matching words and (2) to determine the n best matching words from a set of words and categories proposed by the Syntactic-Semantic component (here labeled "Parser") as being acceptable to the pragmatic grammar, starting or ending on a given set of word matches in the utterance. Though it is called often in this latter capacity, the amount of CPU time it spends in processing such a set is minimal [see Section II.B.].

The Verifier component makes use of generative acoustic-phonetic knowledge, to validate a sequence of phonemes against a portion of the parametric representation of an utterance. Because it operates on a different representation of an utterance than the Lexical Retrieval component and in a different mode (i.e., synthesizing an ideal parametric model of a phoneme sequence and computing an error metric), it can provide an independent evaluation of a word match or sequence of word matches, with or without context. We have tried several methods of integrating the Verifier into the entire system, all of which have taken into account the Verifier's slowness at scoring word matches and the incommensurability of its scores with those of Lexical Retrieval. Although useful at times, none of the methods yet tried has improved the overall performance of the system. We are currently at work at getting the Verifier to reinterpret its score for a word match such that it can be combined with that of Lexical Retrieval for a

total likelihood measure of correctness. Until that is completed and the Verifier's speed is improved, the Verifier component will remain unaccessed by the system [Cook, 1976].

The Parser (syntactic/semantic KS) performs three functions in the current version of HWIM: (1) to predict the possible extensions of a given partial theory into an acceptable theory which spans the utterance; (2) to judge the grammaticality of a given theory (This is necessary since the parser, to save time, does not perform all its tests on grammaticality before making its proposals); and (3) to produce a parse tree for input to our semantic interpreter. Essentially, our entire strategy for understanding an utterance consists of calls to the parser to evaluate and propose extensions of a given partial theory, interleaved with calls to lexical retrieval (and the Verifier) to match the proposals, the decision as to what partial theory to consider next being made by the speech understanding controller based on a "short-fall" algorithm developed by Woods [see Section II.C.].

The Semantic Interpreter component is currently accessed only after an acceptable spanning utterance has been found and parsed, in order to produce a formal meaning representation of the utterance. This representation, in a language akin to predicate calculus, is then manipulated by the TRIP component in its information retrieval role to

yield a correct response to the utterance. The following are some examples of such formal meaning representations:

```
Utterance:      How many people went to California in November?
Interpretation: (FOR THE X1/ (SETOF X2/ (PEOPLE): (GO (TRAVELER X2)
                  (DESTINATION '(STATE CA)) (TIME '(MONTH NOVEMBER))));
                  T; (OUTPUT (CARDINALITY X1]

Utterance:      Cancel Lyn's trip to Pittsburgh.
Interpretation: (FOR THE X1/(TRIP (TRAVELER '(FIRSTNAME LYN))
                      (DESTINATION '(CITY PITTSBURGH))):
                  T; (CANCEL X1]

Utterance:      Give me a list of untaken trips.
Interpretation: (FOR EVERY X1/ (TRIP (MODALITY 'UNTAKEN)): T;
                  (OUTPUT X1]
```

We have not yet tried to use the semantic interpreter as part of the understanding process. Though we believe it is possible to build some pieces of the final meaning representation during parsing, it may not be as efficient as it seems on paper, due to the wasted cost of interpreting partial parses which might be either discarded or part of a theory that turns out to be spurious. (Such was the case in BBN's LUNAR system [Woods et al., 1972] where we tried and then rejected incorporating semantic interpretation into LUNAR's ATN grammar.)

In addition to embodying both of HWIM's discourse-level and factual knowledge sources, the TRIP component answers queries, carries out computations requested by the manager, and maintains the factual data base. Once an utterance has been understood and processed, the TRIP component generates

a reply [Cook, Bruce, and Gould, 1975] in English. The text reply can be printed out to the manager or it can be converted into a phonemic representation with prosodic cues specified. In the latter case, TALKER is called to generate a speech waveform which can be played out to the manager as recognizable speech.

To summarize, the function of each of HWIM's knowledge sources is both well-defined and limited, as opposed to each KS offering up for grabs every piece of information it is capable of producing. As we have tried to indicate in this paper, we do not believe in the latter approach to system organization. Using the knowledge sources as described above, with the decision as to which theory to pursue based on the "short-fall" algorithm referenced earlier, our success rate in recognizing new utterances has been running approximately 20%.

5) Future Changes to HWIM

Currently in HWIM, the knowledge for semantic interpretation of input sentences exists in a separate component from the one embodying the pragmatic grammar. This organization is satisfactory when the semantic interpreter is used merely as a second stage processor that builds formal meaning representations of utterances from their parse trees. However, another important use can be

made of the interpreter. If interpretations are made for major constituents during the understanding phase itself, then it becomes possible to verify that the constituent is not only meaningful in the travel budget domain, but also in the context of the then-current state of the factual data base and the discourse. All this implies a close coupling of the semantic interpretation KS with the syntax/semantics KS. Such a coupling is now being considered. Although it has the additional side benefit of making it easier for the interpreter to take advantage of local data structures (e.g., registers in the grammar) established by the syntax/semantics KS during parsing, its total benefit to the entire system is not necessarily guaranteed.

Another imminent change to HWIM involves re-integrating the Verifier component into the rest of the system, as mentioned earlier. Basically, we are considering methods of combining scores from the Lexical Retrieval component with those of the Verifier component. The method of combination must be based upon considerations of where each is most effective and what impact both correct and incorrect results would have on overall system performance.

For example, one purpose for consulting either KS is to get an evaluation of the goodness of a word match. For long words and phrases the Verifier is very effective since it is able to access speech input data at the parametric level and

can measure precisely the fit of the hypothesized word(s) to the waveform. The Verifier's score is even more reliable when it can use the phonetic context in making this fit. If the context is not specified or is specified incorrectly, then the reliability of the score naturally suffers. For small words these boundary errors are relatively more damaging because there is less in the middle of the word to rectify the score. Phonological variations in the pronunciation of small words also have relatively larger effects. Thus, we cannot place much reliance in the verification score for small words. On the other hand, if the purpose is to check for the mere existence of a small word in the utterance, the Lexical Retrieval component often fails because of segmentation errors. That is, a small word may be subsumed by the start or end of an adjacent word. In that case, Lexical Retrieval will give only a very poor score, while the Verifier may, given the context, find the small word.

In general, these examples show that one must consider the conditions under which a KS is operating and how its output is to be used in order to have a successful system. Once again, we are reminded of common sense principles which run counter to some deceptively attractive proposals regarding system control [McDermott, 1976].

6) Conclusion

The evolution of HWIM discussed in the preceding sections has produced a speech understanding system with much improved performance characteristics. Equally important, though, are the insights into system organization and control which may be gleaned from this history. Below, we list an obviously incomplete set of principles, alluded to in the examples above, which have been implicitly followed in making changes to HWIM.

DON'T DO MORE THAN YOU HAVE TO

> Principle: A KS only provides information to those KS's that can use the information efficiently to improve overall system performance.
>
> > Corollary: If there is no KS that can efficiently use some type of information potentially available from a given KS, the information is not provided and may not even be computed.

WORK TOGETHER

> Principle: If it is found that one must frequently consider simultaneously information from several KS's, then the activity of those KS's should be tightly coupled.

BE AWARE OF YOUR AUDIENCE

> Principle: The form in which a KS packages the information it is asked to provide and the circumstances under which that information will be used depend upon characteristics of the recipient KS.

References

Cook, C., B. Bruce and L. Gould (1975)
"Audio Response Generation," Speech Understanding Systems, Annual Technical Progress Report, 30 October 1974 to 29 October 1975, Report No. 3188, Bolt Beranek and Newman Inc., Cambridge, Mass., pp. 95-110.

Cook, C. (1976)
"Word Verification in a Speech Understanding System," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, April 12-14, Philadelphia, pp. 553-556.

McDermott, D. (1976)
"Artificial Intelligence Meets Natural Stupidity," SIGART, April, pp. 4-9.

Schwartz, R. and V. Zue (1976)
"Acoustic-Phonetic Recognition in BBN SPEECHLIS," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, April 12-14, Philadelphia, pp. 21-24.

Woods, W.A. (1970)
"Transitional Network Grammars for Natural Language Analysis, CACM, Vol. 13, No. 10, October.

Woods, W.A., R. Kaplan, and B. Nash-Webber (1972)
"The Lunar Sciences Natural Language Information System: Final Report," BBN Report No. 2378, Bolt Beranek and Newman Inc., Cambridge, Mass.

Woods, W.A. and J. Makhoul (1974)
"Mechanical Inference Problems in Continuous Speech Understanding," Artificial Intelligence 5, pp. 73-91.

Woods, W.A. and V. Zue (1976)
"Dictionary Expansion via Phonological Rules for a Speech Understanding System," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, April 12-14, Philadelphia, pp. 561-564.

B. Probabilistic Lexical Retrieval with Embedded Phonological Word Boundary Rules

John W. Klovstad

Abstract

This paper discusses the design and realization of a method of lexical retrieval which has been incorporated into the BBN speech understanding system. As a component of this system, Lexical Retrieval operates on a phonetic segment lattice whose segments are described probabilistically. Its function is to return an ordered list of the most probable words and their location within the lattice. The words considered in any such scan can be constrained as to class, location, and phonetic context. This method of lexical retrieval takes advantage of a representation of phonological word boundary rules which enables their effective use, independent of whether the phonetic context is unknown as in word spotting, or known, as in a scan to the left or right of a previously located word. This ability to take account of word boundary effects in a clean and efficient way is one of the major accomplishments of this work.

1) Introduction

The intent of this paper is to describe a method of lexical retrieval and its incorporation as a component of the BBN Speech Understanding System. This development is described from a different perspective in each section of this paper. We begin by considering various criterion which have affected the design of this system of lexical retrieval, either its dictionary structure or the look-up algorithm, and examine how they motivated this design. We also consider the extent to which each criteria could have affected the structural or algorithmic design. Sections 3 and 4 describe in detail the design of the dictionary structure and look-up algorithm respectively. Section 5 discusses lexical retrieval specifically as a component of the BBN Speech Understanding System. Section 6 describes a test that was undertaken to evaluate its performance.

2) Design Motivation

There were five separate criteria taken into account in the design of this Lexical Retrieval system:

1) that it support our scoring philosophy;
2) that it be able to handle expected segmentation errors;
3) that it recognize and allow for word boundary effects;
4) that it be able to perform selective retrieval; and
5) that it do all the above efficiently while maintaining expandability.

These criteria will be explained individually in the following sections, then referenced later in describing the design of the dictionary and the look-up algorithm.

2.a Scoring Philosophy

Our scoring philosophy involves assigning a score Si to pronunciation Pi in proportion to the probability of Pi given the acoustic input A (i.e., P(Pi|A)). Selecting the "best" scoring pronunciation is then equivalent to selecting the most probable pronunciation given the acoustic evidence.

Using Bayes rule, we can rewrite P(Pi|A), which we cannot compute directly, as P(Pi)*P(A|Pi)/P(A). Although an estimate of P(Pi) is associated with each pronunciation only P(A|Pi)/P(A) is computed to rank them. Note that if two pronunciations being compared both span the same acoustics A, their ranking will depend only on the calculation of P(A|Pi). In practice, however, the acoustics spanned by the pronunciations are different and the whole expression must be used. This ratio is commonly referred to as the likelihood ratio.

Since Pi is not in general a single phoneme but rather a specific phoneme sequence, the question of how P(A|Pi) should be evaluated must be answered. Our Lexical Retrieval system does not work off the raw acoustics of an utterance but rather a lattice representation in which the utterance

is segmented into labelled regions. Each segment has a probabilistic descriptor which contains for every phoneme an estimate of the probability of its underlying the region. An expansion of P(A|Pi) then requires establishing a correspondence between phonemes in Pi and segments associated with the acoustics A. These segments must be adjacent and follow in the same order that the phonemes do in the pronunciation Pi. Furthermore the contextual effect of surrounding phonemes must be known for the scoring of each phoneme-segment combination. A pronunciation likelihood is computed by multipling the likelihoods of each phoneme-segment correspondence.

The log likelihood ratio (the logarithm of the likelihood ratio) will rank word matches in the same order as the likelihood ratio. This is true because the logarithm of a monotonic function is monotonic. Since log likelihocd ratios compress the dynamic range and can be calculated by summing, slow floating point multiplication (relative to integer addition) is unnecessary. Therefore Lexical Retrieval calculates log likelihood ratios to boost computational efficiency.

The effect of scoring philosophy on the design of Lexical Retrieval has been to require individual phonemes (allophones if there are contextual effects) to be directly accessible from the structural representation of the

dictionary. This is a very weak constraint and could have been satisfied by many different structures.

2.b Segmentation Problems

If the acoustic segmenter is only permitted to create a single linear sequence of segments (as will be our initial assumption), then either its missing a real boundary or its finding an imaginary boundary will affect the alignment situation. We cannot count on a one-to-one alignment of each phoneme in the pronunciation and segment in the segment lattice. Although the segmenter sometimes misplaces boundaries as well, phoneme-segment alignment is rarely affected since the labels assigned to those segments do not change significantly.

Alignment can be further complicated by multiple consecutive segmentation errors of the same type. Fortunately such multiple consecutive segmentation errors occur very infrequently and no special procedure has been adopted to handle them. Therefore, when they do occur, a "correct" alignment is impossible, although a good scoring alignment may often be obtained. For this reason, the extra computational effort required to permit correct alignments does not appear to be justified and they will not be handled or discussed further in this paper.

The effect on the design of Lexical Retrieval of having to consider segmentation errors is primarily algorithmic. That is, we have to consider all possible alignments in scoring a pronunciation against the acoustic representation.

2.c Word Boundary Effects

In continuous speech, word pronunciations show strong context-sensitive effects, especially at their boundaries. Tokens of short words may bear only a slight resemblance to their counterparts spoken in isolation. For example, the sequence "did you" in continuous speech may become "dijew" when in fact neither "di","dij","jew" or "ew" are acceptable pronunciations for "did" and "you" spoken in isolation.

The need to accommodate for context-sensitive word boundary effects, especially when the context may not have yet been determined, imposes stringent constraints on effective dictionary structures, as will be shown in section 3.c.

2.d Selective Retrieval

As a component of HWIM [see Section II.A.], Lexical Retrieval is meant to work in a mode in which it responds to proposals made by other components. A proposal is a declaration of anticipated words and is often made in terms of fixed classes (e.g., syntax classes). Therefore, it is

desirable that Lexical Retrieval be able to match selectively words belonging to specified classes.

2.e Efficiency and Expandability

The fifth criterion motivating the design of our system for lexical retrieval is efficiency and expandability. What is desired is a structure that is reasonably efficient in the use of memory - so that the structure is feasible for use with very large dictionaries - and at the same time admits an algorithm which is reasonably efficient - so that the scoring technique is still feasible.

2.f Summary

Of the five design factors mentioned above, all but the need to accommodate for word boundary effects lend themselves to incorporation in a wide variety of radically different internal dictionary structures. Consequently, the design of the dictionary primarily depends on the "solution" of the so-called word boundary problem. In the following section, some effort will be made to indicate the nature of this problem and how its solution suggests the design of the particular structure we have chosen.

All five design factors have had some effect on the look-up algorithm which complements our chosen dictionary structure. Section 4 describes the look-up algorithm in

considerable detail, noting the influence of each design factor.

3) Internal Dictionary Structure

3.a Word Boundary Rules

We characterize context-sensitive word boundary effects in terms of phonological word boundary rules. Each rule typically specifies first, some phonetic context in which an underlying phonetic sequence may be expected to change, and second, the phonetic sequence expected. The contextual portion of word boundary rules specifies the general phonetic character of the word boundary at which the changed phonetic sequence may be observed. The word boundary effects dealt with in the paper will all be specified in the following formalism:

Left Context # Right Context => Transformation

where # designates the word boundary. A typical rule would be:

T # T => T

which states that if a word ending in T (Left Context is T) is followed by a word beginning with T (Right Context is T), then an optional effect is that only one T will be observed. It was decided to make rule application optional in order to

reduce complexity.

The reason why the word boundary problem is seen to significantly constrain the dictionary structure is the following. Suppose for the sake of this example that words are matched from left to right. Consider any word boundary rule specified in the aforementioned formalism. The set of words that satisfy its right context are precisely those words that start with the specified phonetic sequence. An efficient way to represent such a set is in a tree structure which merges common initial sequences. The set of words is then uniquely specified by the node in the tree which terminates the right context. Thus, given left-to-right matching, the representation of word boundary rules strongly suggests the dictionary be organized as a right branching tree structure.

3.b Trees - Definition and Representation

A tree consists of a set of nodes connected by branches. The direction of node connection, although not indicated, is consistently left-to-right. A tree is further specified by the following conditions:

1) It contains exactly one node (the root) which no branch enters.
2) It is connected so that for each node there is a path (sequence of branches) from the root to that node.
3) Each node except the root has exactly one branch that enters it.

In each of our figures a node will be represented by a circle, usually labeled with a single phonetic symbol. Thus each left-to-right path in the tree is associated with a single phonetic sequence or pronunciation.

3.c Dictionary "Trees"

The following subsections will describe and illustrate a sequence of progressively more complex structures for representing dictionary information:

1) the basic tree
2) the kernel tree
3) the kernel tree with embedded word boundary rules.

Section 3.c.1 illustrates a tree constructed from a specific set of pronunciations. It indicates how with minor additions this structure can be made to satisfy design constraints 1, 4 and 5 mentioned in section 2. Section 3.c.2 introduces the concept of pronunciation kernels relative to a given set of word boundary rules and illustrates the corresponding kernel tree. Section 3.c.3 illustrates the designed structure in which paths appropriate for the specified word boundary rules have been added to the kernel tree. Together, the last two subsections indicate how further modifications to this structure can be made to satisfy constraint 3.

3.c.1 Basic Tree

The basic pronunciation tree is formed by merging those common initial phoneme sequences of word pronunciations such that each path in the tree specifies a unique phonetic sequence. As a result, no two distinct paths through the tree are associated with the same phonetic sequence, and no two distinct initial phonetic sequences are associated with the same path.

From the words and associated pronunciations given in Figure 1, we can construct the basic tree shown in Figure 2. Notice that words are identified at the node in the tree which complete the path associated with its pronunciation. A pronunciation path is a path that terminates on a node in the tree at which some word is identified. The set of all sequences of pronunciations can now be generated by the concatenation of appropriate pronunciation paths.

That each node in the tree (except the root) is labeled with a single phonetic symbol satisfies the constraint imposed by the scoring philosophy that individual phonemes (allophones if there are contextual effects) be directly accessible from the structural representation of the dictionary.

To satisfy our desire for a selective retrieval capability, each node can be tagged with the subset of

LEXICAL	PHONEMIC
HAD	h æ d
HAND	h æ n d
YOU	y u
AND	æ n d
ANALYZE	æ n ! ɑy z
ANSWERS	æ n s ɝ z
LAST	l æ s t
LABELS	l e b ! z
LIST	l I s t
LIKELY	l ɑy k l i
SENT	s ɛ n t
SYMBOL	s I m b ə l
SCOPE	s k o p
ALL	ɔ l
ON	ɔ n
THE	ð ə
THIS	ð I s

Fig. 1. Vocabulary

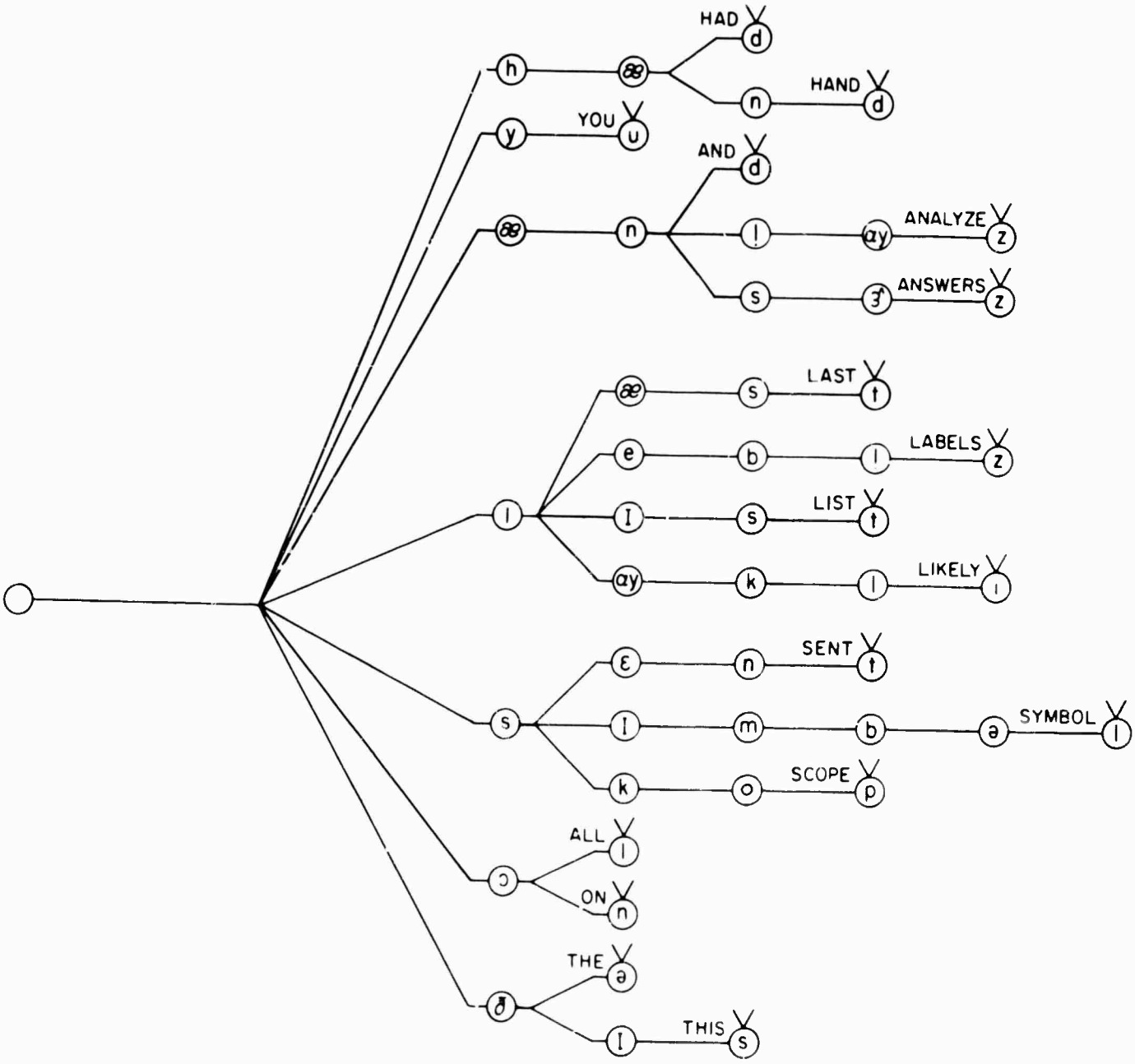


Fig. 2. Basic tree.

syntax classes which are instantiated by one or more of the words whose pronunciations "pass through this node". During selective retrieval, entire sub-trees can be eliminated from consideration if the requested syntax classes cannot be instantiated.

That the amount of memory required to represent pronunciations in a tree structure is reasonable is seen by counting the number of nodes in such a tree. If the pronunciation information were not shared, the amount of storage would be proportional to the total number of phonemes in the pronunciations. With any sharing, fewer nodes will be needed, though the number will be at least as great as the number of distinct pronunciations, since each has a corresponding terminal node. Fortunately, as more and more words are added to the dictionary tree, there is more and more merging of paths near the tree root. Although this saving in storage may not seem significant, given the tree shown in Figure 2, it becomes evident for a large vocabulary, as illustrated in Figure 3. Thus, sharing common initial subparts has benefits for both storage and retrieval.

	A	B	C	B/A	C/A
Vocabulary Size	Number of Pronunciations	Total Nodes	Total Phonemes	Nodes per Pronunciation	Phonemes per Pronunciation
Small	17	50	63	2.941	3.705
Large	1860	4371	10616	2.350	5.707

Figure 3.

3.c.2 Kernel Tree

The tree described above, although satisfying the constraints of scoring philosophy, selective retrieval and efficiency, does not accommodate word boundary effects. This requires a slightly different tree which we will call a Kernel tree, because instead of representing word pronunciations as the basic tree, it represents their kernels.

The kernel (K) of a pronunciation is defined with respect to a given set of phonological word boundary rules. If P is a phonetic spelling and R, the set of word boundary rules whose left context is satisfied by P, then the kernel K is defined as the longest left substring of P which would be unchanged by the application of any of the rules R.

The Kernel tree for the vocabulary in Figure 1 relative to the rules given in Figure 4 is shown in Figure 5. As an example, notice that the kernel of L I s t is L I s because of the rule s t # t => s. The other rule whose left context is satisfied by L I s t is t # th => th, but application of this rule will not change the kernel L I s.

Fig. 4. ⟶

RULES

n d # l → n l
s t # s → s
d # y → ǰ
s # s → s
l # l → l
t # ð → ð

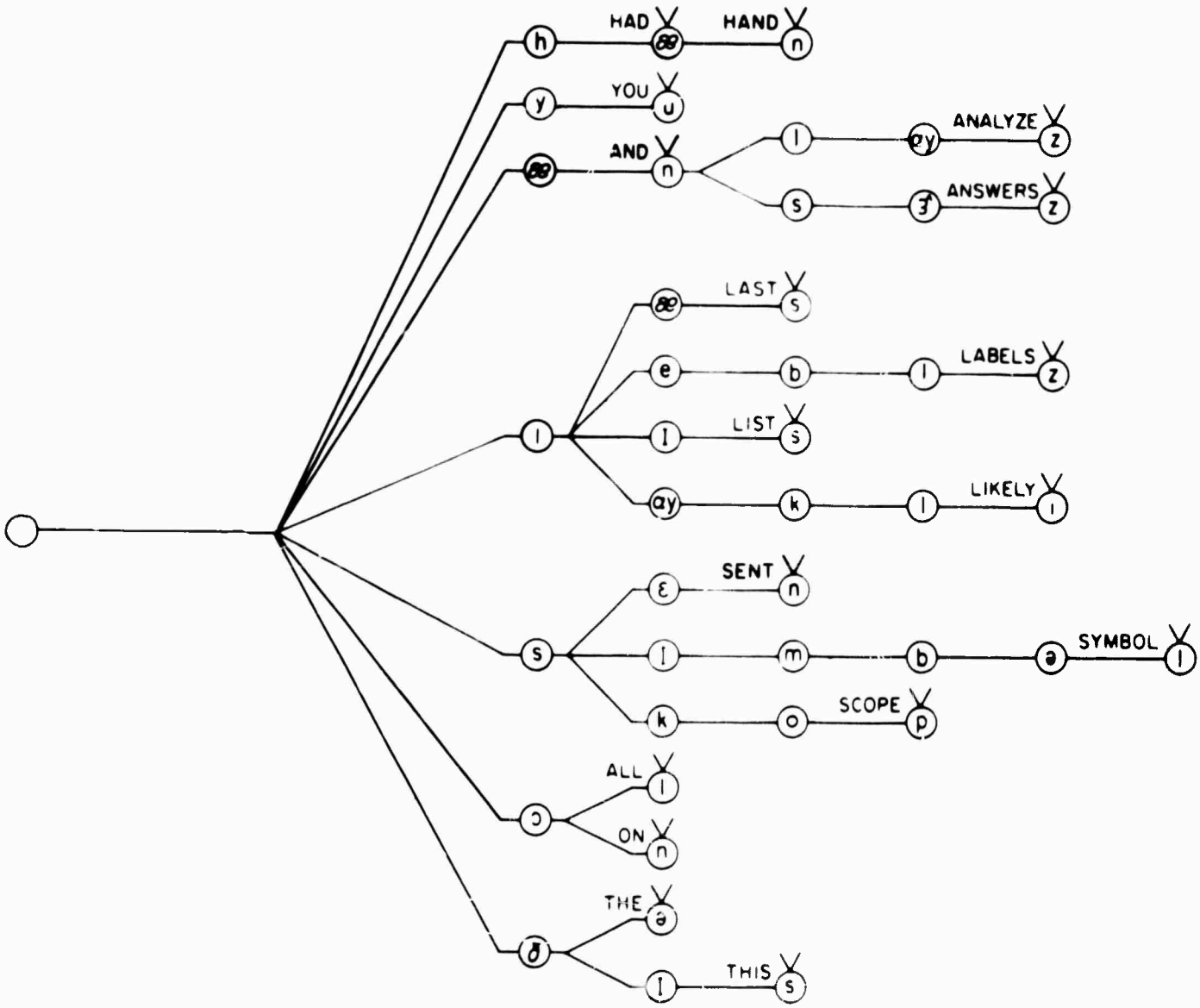


Fig. 5. Kernel tree.

3.c.3 Kernel Tree with Embedded Word Boundary Rules

Each kernel satisfies the left context of some subset (maybe empty) of the given word boundary rules. Both kernel and subset are remembered and a correspondence between them is established. After the Kernel tree has been completed, additional paths must be added to permit the transformations specified in the rules and to complete the kernels (if necessary). This is accomplished by constructing an entry tree for each distinct rule subset and kernel extension.

One path in each entry tree provides for the completion of the kernel. At the node terminating this path there is a branch that enters the root of the Kernel tree. The other paths provide for the transformations in the rule subset. At each node that completes a transformation there is a branch that enters the Kernel tree at a node determined by the right context of the corresponding rule.

The structure that results after these additions have been made is illustrated in Figure 6. The branches that leave terminal nodes in entry trees and enter the Kernel tree are dashed for clarity. An entry index above the terminal node of each kernel specifies its corresponding entry tree. Now by concatenation of these pronunciations - starting a new path at the tree that corresponds to the entry index associated with the word in the Kernel tree - one observes that all sequences of pronunciations and

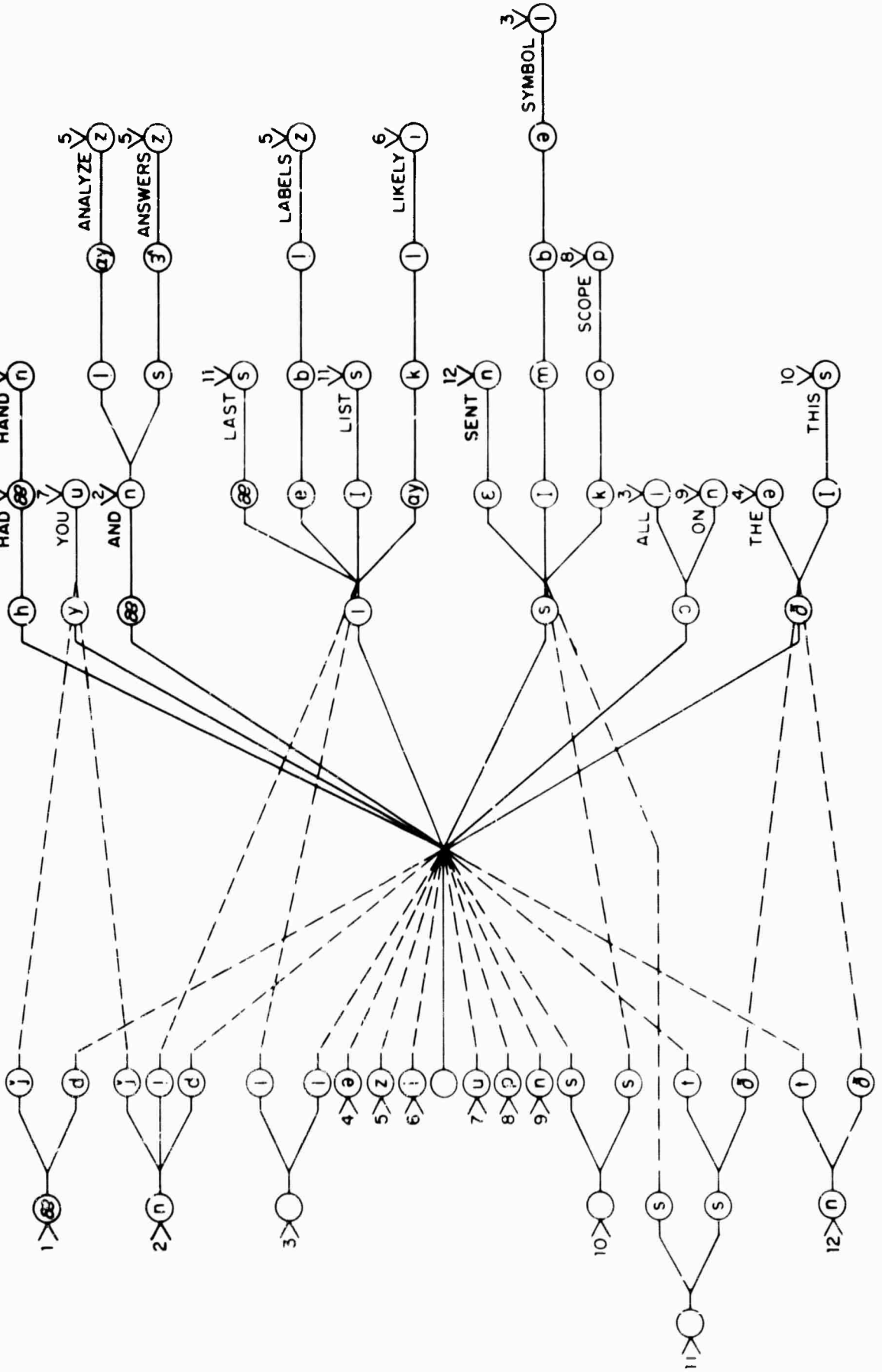


Figure 6. Kernel tree and embedded rules.

possible word boundary effects can be generated.

4) Algorithm

In this section we will describe the look-up algorithm involved in Lexical Retrieval. In addition to the dictionary structure, this algorithm accesses two identical stacks in which path, alignment and score information is temporarily stored. In this description one stack will be called the current path stack and the other the temporary path stack. The details of the algorithm discussed in subsequent sections follow the basic steps outlined here.

As an initialization step, a single path is placed in the current path stack. This initial path is unique in that it: 1) starts and ends at the tree root, 2) is aligned against a segment sequence of zero length, and 3) has a score of zero.

Each path alignment - the alignment of a path against a segment sequence - in the current path stack is extended to include the next segment. Several new path alignments may be generated from each one in the current path stack because of branching in the tree structure and attempting to correct for possible segmentation errors. Each new path alignment has its own score and is stored in the temporary path stack. When each of the path alignments originally on the current path stack has been extended to include the current segment,

the two stacks pointers are interchanged, the new current path stack becoming the old temporary path stack. The next segment is selected, the new temporary path stack is cleared, and this sequence of operations is repeated until all paths in the tree have been followed to completion.

4.a Path Alignment

In this section we will describe the incremental alignment of a single path, the implementation which permits this to be done efficiently, and the combinatoric growth of alignment possibilities. We conclude by extending this procedure to include simultaneous alignments of paths in a tree structure.

All permitted path alignments are generated by a unique sequence of "indivisible" incremental alignments. Three kinds of incremental alignments - "match", "merge" and "split" - are described here. A match refers to the incremental alignment of a single phoneme with a single segment. A merge refers to the alignment of a single phoneme with two adjacent segments. (The segments are said to be merged by this alignment). A split refers to the alignment of two adjacent phonemes with a single segment. (The segment is said to be split by this alignment). Although many other kinds of incremental alignments are conceivable, these three permit sufficient alignment

flexibility to compensate for the most frequently occurring types of segmentation errors.

The only constraint in the generation of path alignments from incremental alignments is that each phoneme in the path and segment in the lattice may participate in only a single incremental alignment. Figure 7 illustrates in terms of incremental alignments all possible alignments of up to length 4. In this figure, incremental alignments are represented symbolically: "I", "/\", and "\/" for match, merge, and split, respectively. The alignments are pictured as nodes in a tree since alignments of length n are generated incrementally by concatenating one additional incremental alignment to its predecessor's alignment of length n-1. The alignment generated can alter its predecessor's alignment only if the last phoneme was aligned in a match. In this case the last phoneme may be realigned as a merge.

Since only three distinct incremental alignments are permitted, incremental extensions can be made in only three ways. That this is true can easily be seen by observing the relationship between any alignment and those generated from it (see Figure 7). Rather than label each path with its alignment or even its last incremental alignment (which is all that is necessary), we preserve this distinction by keeping the paths in three different stacks. This is done

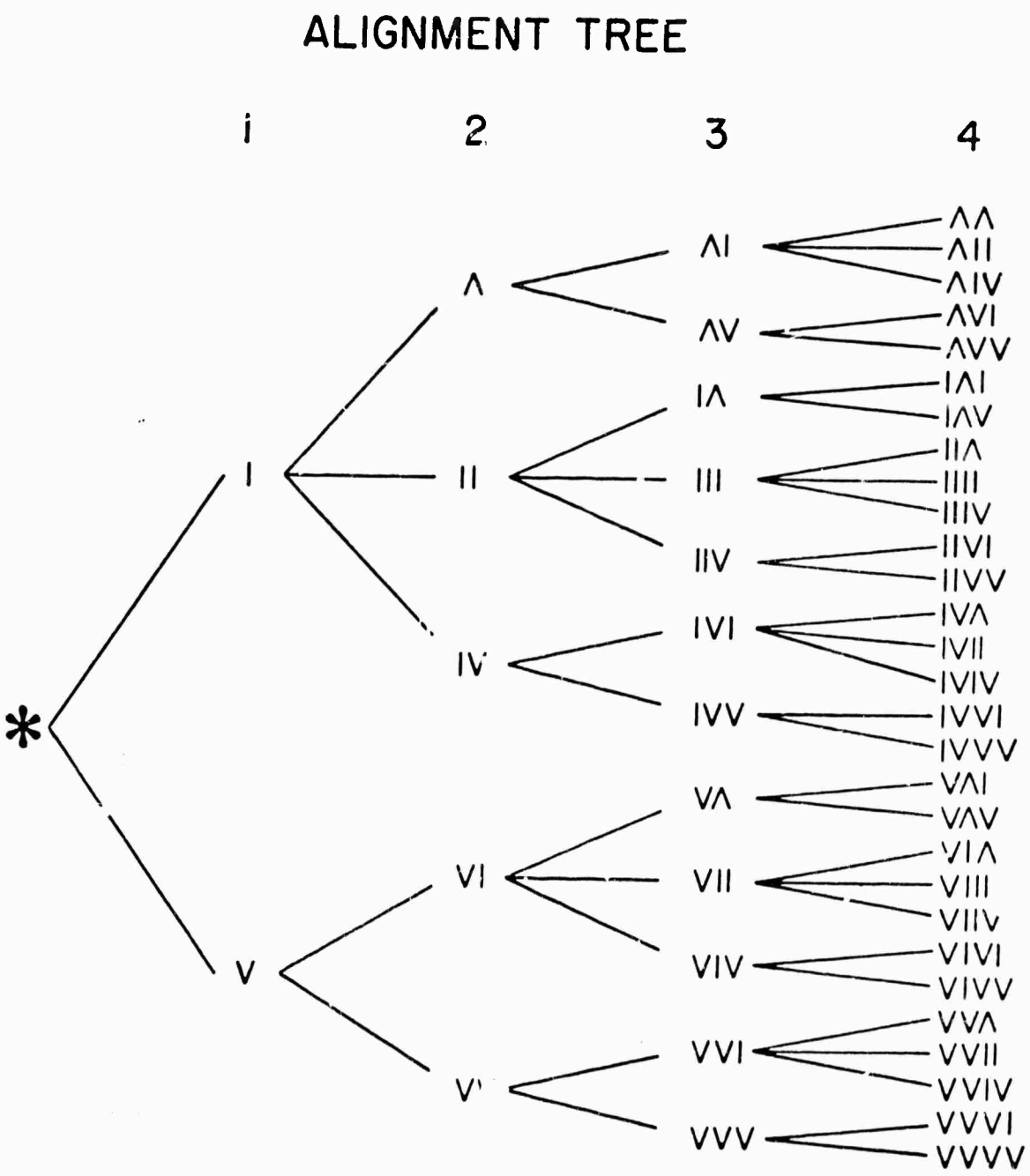

ALIGNMENT TREE
1
2
3
4
*
I
V
Λ
II
IV
VI
V'
ΛI
ΛV
IΛ
III
IIV
IVI
IVV
VΛ
VII
VIV
VVI
VVV
ΛΛ
ΛII
ΛIV
ΛVI
ΛVV
IΛI
IΛV
IIΛ
IIII
IIIV
IIVI
IIVV
IVΛ
IVII
IVIV
IVVI
IVVV
VΛI
VΛV
VIΛ
VIII
VIIV
VIVI
VIVV
VVΛ
VVII
VVIV
VVVI
VVVV

Figure 7.

primarily for efficiency but also because greater control of alignment generation is possible due to individual control of each stack's maximum length (see Section 4.d.3).

The illustration in Figure 8 shows schematically how these alignments are actually generated. There are three stacks for current and temporary path memory instead of just one as indicated in the overview. The three stacks for the current paths are pictured directly above those temporary paths which are being generated. The temporary stacks in the figure are further partitioned - horizontally - to indicate from which of the current stacks their predecessor's alignments derived. The correspondence of descending partitions is with the split, match, and merge stacks respectively.

From this figure, difference equations which relate the number of alignments in each of the three incremental alignment stacks as a function of the number of segments have been written and solved. The results of this analysis is shown in Figure 9 to illustrate the combinatorics involved in the general alignment problem for a single path and a linear segment sequence.

This technique of extending alignments incrementally is readily generalizable to multiple paths, as in the described tree structure. The only difference is that the path alignment stacks will fill up much more rapidly because of

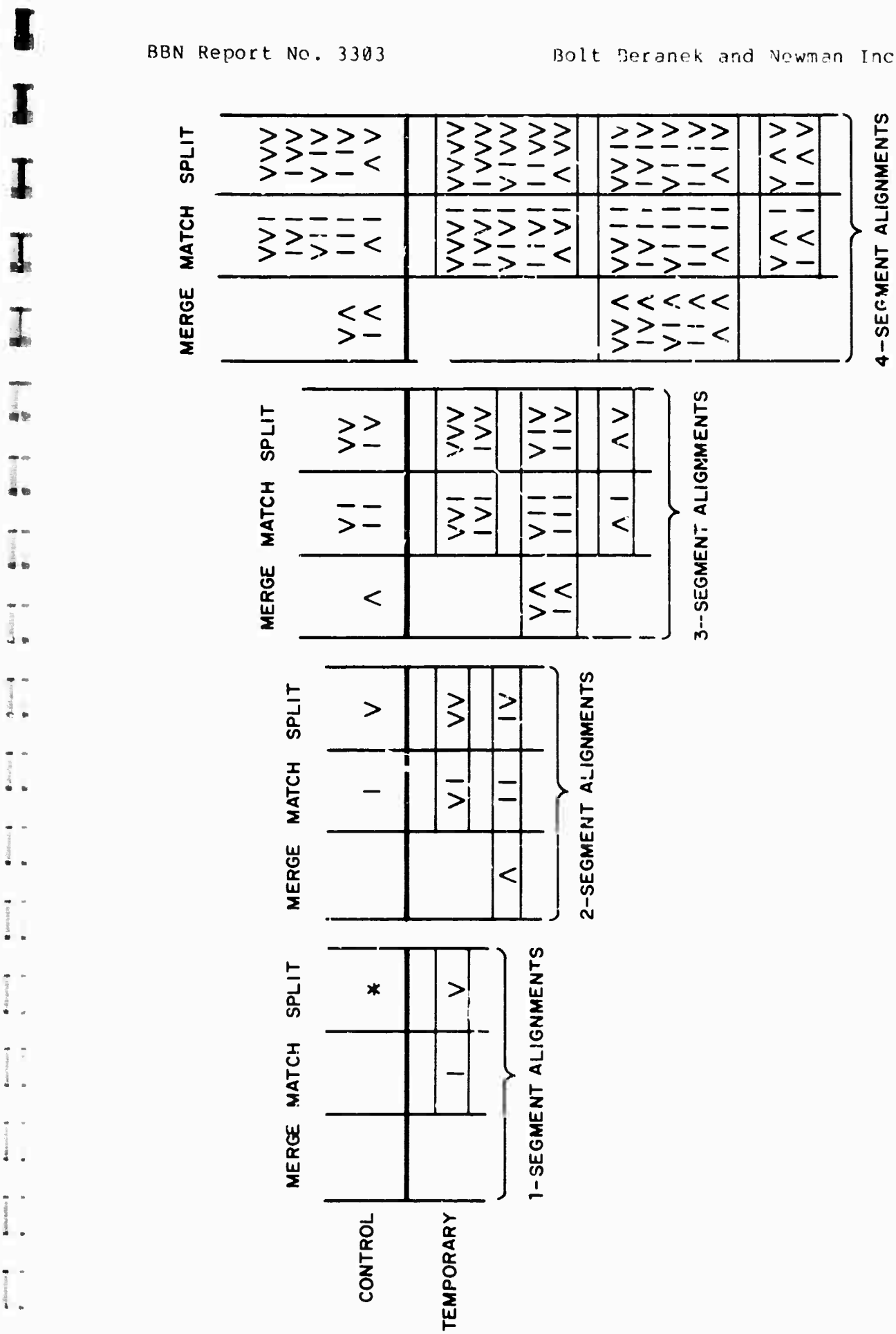


Figure 8. Alignments & Path Extension.

ALIGNMENT COMBINATORICS

t_N = Number of distinct alignments against N segments

$$C_N = \sum_{i=1}^{N} t_i$$

= Number of operations necessary to generate all alignments against N segments (generates all alignments against shorter left sub-sequences of segments as a by-product)

Alignment Description	Match	Match & Merge	Match & Split	Match & Merge & Split
Difference equation	$t_N = MATCH_N$ $MATCH_N = t_{N-1}$ $t_N = t_{N-1}$	$t_N = MERGE_N + MATCH_N$ $MATCH_N = t_{N-1}$ $MERGE_N = MATCH_{N-1} = t_{N-2}$ $t_N = t_{N-1} + t_{N-2}$	$t_N = MATCH_N + SPLIT_N$ $SPLIT_N = MATCH_N$ $MATCH_N = t_{N-1}$ $t_N = 2t_{N-1}$	$t_N = MERGE_N + MATCH_N + SPLIT_N$ $SPLIT_N = MATCH_N$ $MATCH_N = t_{N-1}$ $MERGE_N = MATCH_{N-1} = t_{N-2}$ $t_N = 2t_{N-1} + t_{N-2}$
Initial conditions	$t_o = 1,\ t_1 = 1$	$t_o = 1,\ t_1 = 1$	$t_o = 1,\ t_1 = 2$	$t_o = 1,\ t_1 = 2$
Number of alignments	$t_N = 1$	$t_N = \left(\frac{5+\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^N + \left(\frac{5-\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^N$	$t_N = 2^N$	$t_N = \left(\frac{2+\sqrt{2}}{4}\right)\left(1+\sqrt{2}\right)^N + \left(\frac{2-\sqrt{2}}{4}\right)\left(1-\sqrt{2}\right)^N$
Number of operations	$C_N = N$	$C_N = \left(\frac{5+3\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^{N+1} + \left(\frac{5-3\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^{N+1} - 2$	$C_N = 2^{N+1} - 2$	$C_N = \left(\frac{1+\sqrt{2}}{4}\right)\left(1+\sqrt{2}\right)^{N+1} + \left(\frac{1-\sqrt{2}}{4}\right)\left(1-\sqrt{2}\right)^{N+1} - \frac{3}{2}$

Asymptotic Behavior

Number of alignments	$t_N \to 1$	$t_N \to .7236 * 1.618^N$	$t_N \to 2^N$	$t_N \to .8536 * 2.4142^N$
Number of operations	$C_N \to N$	$C_N \to 1.1708 * 1.618^{N+1} - 2$	$C_N \to 2^{N+1} - 2$	$C_N \to .6036 * 2.4142^{N+1} - 1.5$

Figure 9.

multiple paths and branching in the tree. Only if alignments in the current match stack are being merged will the number of alignments necessarily remain the same.

4.b Path Scoring

The scoring philosophy indicates that the score for a particular alignment is the product of the incremental alignment scores. While the incremental score due to a match can be closely approximated, accurate score approximation for split and merge alignments is more difficult. The difficultly arises because of the infrequent occurrences of split and merge alignments in our limited data base. Therefore, the score actually calculated for either a split or merge is really just a combination of two scores - one for each phoneme-segment combination - and an empirically determined penalty. This is easily computed yet produce a workable approximation to the "correct" score.

4.c Word Completion

As paths are being extended to create new paths, terminal nodes will be encountered corresponding to complete word pronunciations. Since only the best word matches are to be saved on a completion stack, additional restrictions which serve to define the concept of a "best match" are provided by two control parameters: the minimum acceptable

word match score and the maximum number of word matches. These parameters will be called the threshold and completion stack length, respectively.

Word matches are saved on the completion stack in order of decreasing score if their score exceeds this threshold. Whenever the completion stack is full, the threshold "floats" to the score of the worst scoring word match on the stack. Further addition of word matches to the completion stack will bump the worst word match and cause the threshold to "float" higher to equal the score of the new worst word match on the stack. The most probable word matches will be on the completion stack by the time the cycle is completed.

4.d Path Elimination

The previous sections have all treated every path as potentially extendable. This is not the case in actual operation. To reduce storage and computation demands paths are systematically eliminated as soon as possible by three completely different operations: selection, pruning and remembering with finite memory.

4.d.1 Selection

Paths are eliminated by selection if none of the words reachable from the current node belong to previously selected classes. This procedure is always safe because it

only removes from consideration ultimately inappropriate words.

4.d.2 Pruning

Paths are eliminated by pruning if they are unlikely to yield good scoring words. This can be operated in two modes:

1) Immediate pruning - A path is eliminated if its score is less than the current threshold. This is only safe if incremental scores are known to be zero or negative, thus implying a monotonically decreasing word score.

2) Potential pruning - A path is eliminated if its score plus its potential score is less than the current threshold. The potential score of a path is the maximum score possible over its complete extensions. It is computed prior to any scans and tagged to that node. This operation is guaranteed not to throw away any path which could yield words good enough to get on the completion stack.

4.d.3 Remembering with a Finite Memory

In the same way that one can specify the maximum number of words allowable on the completion stack, so can one specify the maximum length of each of the merge, match, and split stack. Given a big enough vocabulary, these stacks will tend to fill up during the look-up operation, necessitating the elimination of some paths, namely the worst of the currently active ones. Although this is not guaranteed to be safe, the probability of a good word being eliminated can be made arbitrarily small by increasing the

maximum length of these stacks.

5) Enhancements

This section discusses five areas in which this method of lexical retrieval has been extended to incorporate it within the BBN Speech Understanding System. These enhancements were implemented so that all of the capabilities just described could be maintained.

5.a Segment Lattice

Lexical Retrieval has been generalized to accept an arbitrary segment lattice as input. (As mentioned earlier in the alignment description, we had been assuming a linear segment sequence). To do this, we arrange the segments in order of increasing left boundary time, assign a unique stack to each boundary, and then extend paths in a sequence determined by the segment's boundaries.

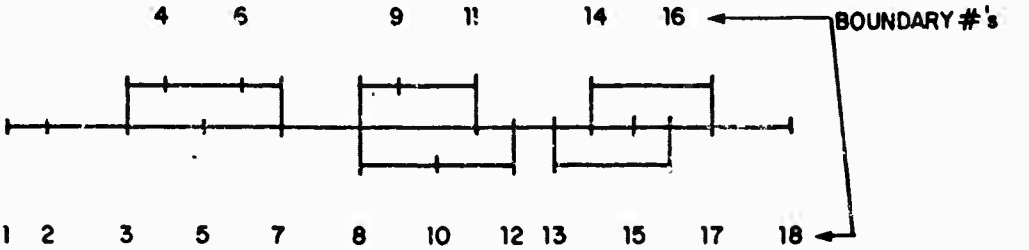
Within this ordering, it is possible for two or more segments to have the same left boundary time. If so, these segments are arranged in order of increasing right boundary times. Figure 10 illustrates an example of this linear ordering of a hypothetical segment lattice. Each boundary in the segment lattice is assigned a stack. Any path which has been aligned with the segment lattice immediately to the left of a given boundary is stored in its assigned stack

SEGMENT ORDERING FOR LATTICE TRAVERSAL

$LB_i \triangleq$ LEFT BOUNDARY # OF i^{th} ORDERED SEGMEN.
$RB_i \triangleq$ RIGHT BOUNDARY # OF i^{th} ORDERED SEGMENT

WHERE BOUNDARY NUMBERS ARE ASSIGNED TO PRESERVE TIME ORDERING.

SAMPLE SEGMENT LATTICE



SEGMENT ORDERING FOR LEFT TO RIGHT TRAVERSAL

i	LB_i	RB_i
1	1	2
2	2	3
3	3	4
4	3	5
5	4	6
6	5	7
7	6	7
8	7	8
9	8	9
10	8	10
11	8	11
12	9	11
13	10	12
14	11	12
15	12	13
16	13	14
17	13	16
18	14	15
19	15	17
20	15	16
21	16	17
22	17	18

ORDERING RULES

LEFT TO RIGHT TRAVERSAL
$LB_i < LB_{i+1}$ OR $LB_i = LB_{i+1}$ AND $RB_i < RB_{i+1}$

RIGHT TO LEFT TRAVERSAL
$RB_i > RB_{i+1}$ OR $RB_i = RB_{i+1}$ AND $LB_i > LB_{i+1}$

SEGMENT ORDERING FOR RIGHT TO LEFT TRAVERSAL

i	RB_i	LB_i
1	18	17
2	17	16
3	17	14
4	16	15
5	16	13
6	15	14
7	14	13
8	13	12
9	12	11
10	12	10
11	11	9
12	11	8
13	10	8
14	9	8
15	8	7
16	7	6
17	7	5
18	6	4
19	5	3
20	4	3
21	3	2
22	2	1

Figure 10.

until paths reaching the boundaries one segment to the right have all been completed.

When traversing the segment lattice, segments are processed in sequence. The paths currently in the stack assigned to the left boundary of each segment are extended to include it and are then stored in the stack assigned to its right boundary.

Once paths at a particular boundary are selected for extension, all segments leaving this boundary must be considered before proceeding to another boundary. If two or more segments have the same left boundary, the stack assigned to the boundary must be preserved until each of the segments have been included.

If two or more segments share the same right boundary, the first segment encountered to reach that boundary must be noticed so that the assigned stack can be cleared. Later, when other segments reference this boundary, their path extensions are added to those already accumulated.

This linear ordering of the lattice permits matching to start on any boundary. The match is started at the first segment to leave the left boundary. If a right boundary is also specified, the match continues until it reaches the last segment to enter the right boundary. During such locally constrained scans, segments whose right boundaries

extend beyond the designated right boundary may be skipped.

The amount of computation is roughly linear to the number of segments in the lattice. Note that although all paths through the lattice are considered, many will be aborted rather quickly by pruning, thus preventing a combinatoric explosion.

5.b Word Selection

It is often convenient when writing grammars to be able to reference single words or lists of words rather than just syntax classes alone. Such is the case in HWIM's pragmatic grammar SMALLGRAM [Woods et al., 1975]. Since Lexical Retrieval must respond to predictions made by HWIM's syntactic component, it must be able to process predictions in which both words and classes are specified. This capability has been included by marking paths in the tree structure that correspond to the pronunciations of the specified words. These paths are followed in addition to those already permitted by class selection.

The marking of paths begins at the terminal node and proceeds towards the root of the kernel tree. Marking terminates whenever a previously marked path is encountered. This capitalizes on the similarity of the different pronunciations of a single word, since each must be marked. Furthermore, by marking the root prior to any path marking,

only a single completion test need be made. Unmarking the tree after each scan is done in exactly the same way.

Besides the advantage of simultaneous class and word selection, each path can take full advantage of the embedded phonological word boundary rules. This implementation of word specification has the disadvantage of a computational overhead, due to the necessity of both marking (and later unmarking) the tree and maintaining only requested paths. However, the continued ease with which word boundary rules can be handled easily outweighs this disadvantage.

5.1 Bi-Direction Lookup

Lexical Retrieval was also extended in HWIM to perform matches both right-to-left as well as left-to-right. This was because the initial path of a correct word would sometimes score poorly against the segment lattice and be eliminated even though the complete path would have scored higher than surviving paths. The assumption motivating this extension was that the ability to look-up words in either direction would preclude such situations.

A further impetus for extending Lexical Retrieval to do bi-directional look-up was a new control strategy adopted in HWIM [see Section II.C.] based on growing word islands outward from highly probable words. This required an ability to do anchored scans off word matches in either

direction, an ability which bi-directional look-up provided.

Conceptually this extension merely required two separate tree structures - one left-going and the other right-going. To maintain storage efficiency, shared information referenced in the two trees was stored so that it could be accessed by word and path indices. Therefore, each terminal node now specifies only word and path indices. The word index references the word's lexical spelling, syntax classes, and all of its pronunciations. The path index references the terminal node and entry node specific to this particular path in the kernel tree. This arrangement establishes a correspondence between paths in the two kernel trees. As expected, words that had previously been missed in one direction because of a poorly scoring initial portion were now found in the other.

5.d Scoring Beyond the Kernel

Additional experience revealed performance deficiencies in the following two situations:

1) A correct but poorly scoring word would be eliminated when this could have been prevented by the high score of some robust phoneme outside its kernel.

2) An incorrect word would outscore correct words when this would not have occurred if the bad score of some robust phoneme outside its kernel had been taken into consideration.

Both problems were observed occasionally and were at least partially responsible for the unsuccessful recognition of some sentences. It was believed that scoring phonemes beyond the end of the kernel would eliminate this particular kind of problem. The Kernel tree with embedded word boundary rules as previously described was used as a guide in the consideration of alternative structures. The structure chosen was based around a kernel tree, though one in which maximal path sharing as previously defined is no longer possible in many situations.

Now when a specially marked terminal node is reached word and path indices are picked up and the path continues as if still in the Kernel tree. At some node in the entry tree a second and different mark would indicate that scoring had proceeded far enough. At unmarked terminal nodes (they end kernel paths which represent complete pronunciations) the path scoring is terminated. As a consequence, two distinct paths may have the same phonetic sequence associated with them. This is because the path that continues into the entry tree appears (except for the mark) as if it is in the Kernel tree, and other paths that would have overlayed it must now be separated. To maximize path sharing, only the terminal nodes of such kernels are made distinct. This new structure is illustrated in Figure 11.

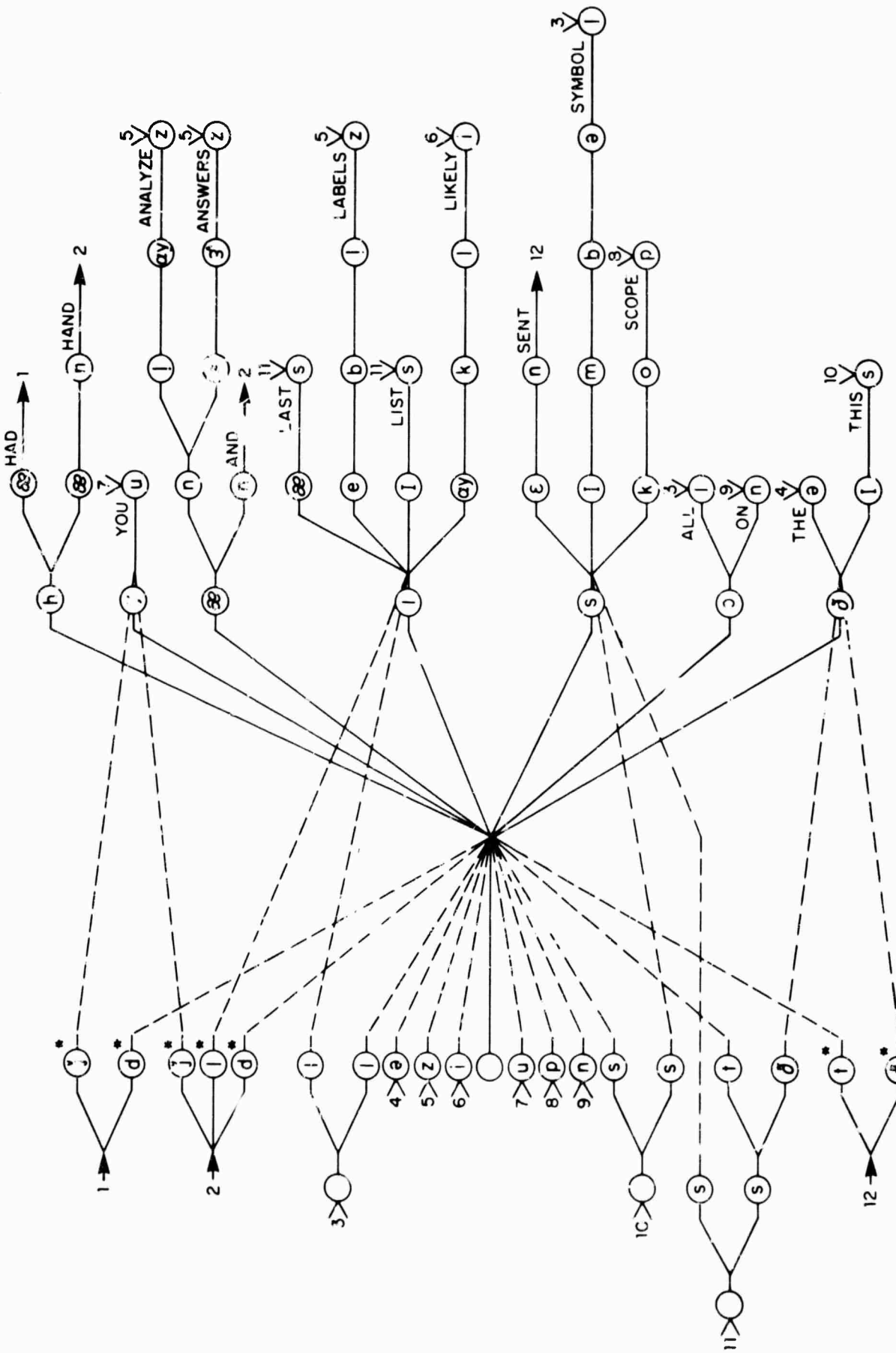


Figure 11. Beyond the Kernel.

5.e Scoring Relative Stress Patterns

Another way in which Lexical Retrieval was extended was to permit relative stress information to be used in computing scores. Previously, the stress had been ignored since phoneme-segment likelihood ratios were based on statistics taken on phonemes with no stress designation. It was felt that the inclusion of stress information would significantly improve the performance of the Lexical Retrieval component. Although the subjective impression of stress is affected by at least amplitude, pitch, and duration, it appeared that estimating stress solely on the basis of amplitude measurements and scoring the relative stress pattern would yield consistent results. We modeled a distribution relating these relative stress differences based on statistics taken to correlate this measure of stress with the stress levels in the speech.

The match score was augmented by a score that was computed for each consecutive vowel pair in the pronunciation and did not modify the score of any monosyllabic word. Each addition to the score was calculated by evaluating the model distribution at a place specified by two relative stress parameters: the difference between the labeled vowel stress and the difference between measured stress on the corresponding "aligned" segments. The look-up algorithm was modified to remember "recent"

stress information (both labeled and measured), calculate stress differences, and evaluate the distribution during path extension.

We found that words with a stress pattern corresponding to the measured stress pattern were definitely favored. Unfortunately, this was true for incorrect words as well as for correct words and there did not seem to be an overall improvement in Lexical Retrieval performance. Therefore the operation of the relative stress scoring technique was conditioned on a flag variable. We have been running the Lexical Retrieval component in a mode where this flag has been turned off. If in the future we are able to measure subjective stress in a manner more consistent with the labeled stress patterns, it may be feasible to attempt the use of relative stress information again. Meanwhile, stress effects that do not depend on relative levels can be accounted for by collecting statistics on the appropriate allophones of each vowel.

6) Performance

Lexical Retrieval with all of the described enhancements is currently functioning as a component in the BBN Speech Understanding System. In order to evaluate its individual performance, it was tested on a set of 99 sentences. These sentences had been recorded and processed

for use with the entire system and constituted (at the time of the test) the entire data base of sentences. Although the number of words per sentence varied considerably, we made no attempt to compensate by requesting a proportional number of word matches. Instead, Lexical Retrieval was directed to return the 15 most probable word matches for each sentence. All words in the dictionary (702) were possible candidates and only correct inflections were counted as correct. Figure 12 shows one performance measure: the average ratio of correct to incorrect words.

Another performance measure: its ability to distinguish between correct and incorrect words is presented in Figure 13. Here the rank of the highest scoring correct word is plotted as a distribution.

In conclusion, we feel we have developed both an efficient and an effective method of lexical retrieval for use in general speech understanding systems. We will continue our effort to develop this component to improve performance, efficiency, and flexibility.

Average Number of Words per Sentence	9.21
Average Number of Distinct Word Matches . .	9.48
Average Number of Correct Words per Scan . .	2.25
Average Number of Incorrect Words per Scan .	7.23
Average Ratio of Correct to Incorrect Words	.3112

Fig. 12. Performance Measurement.

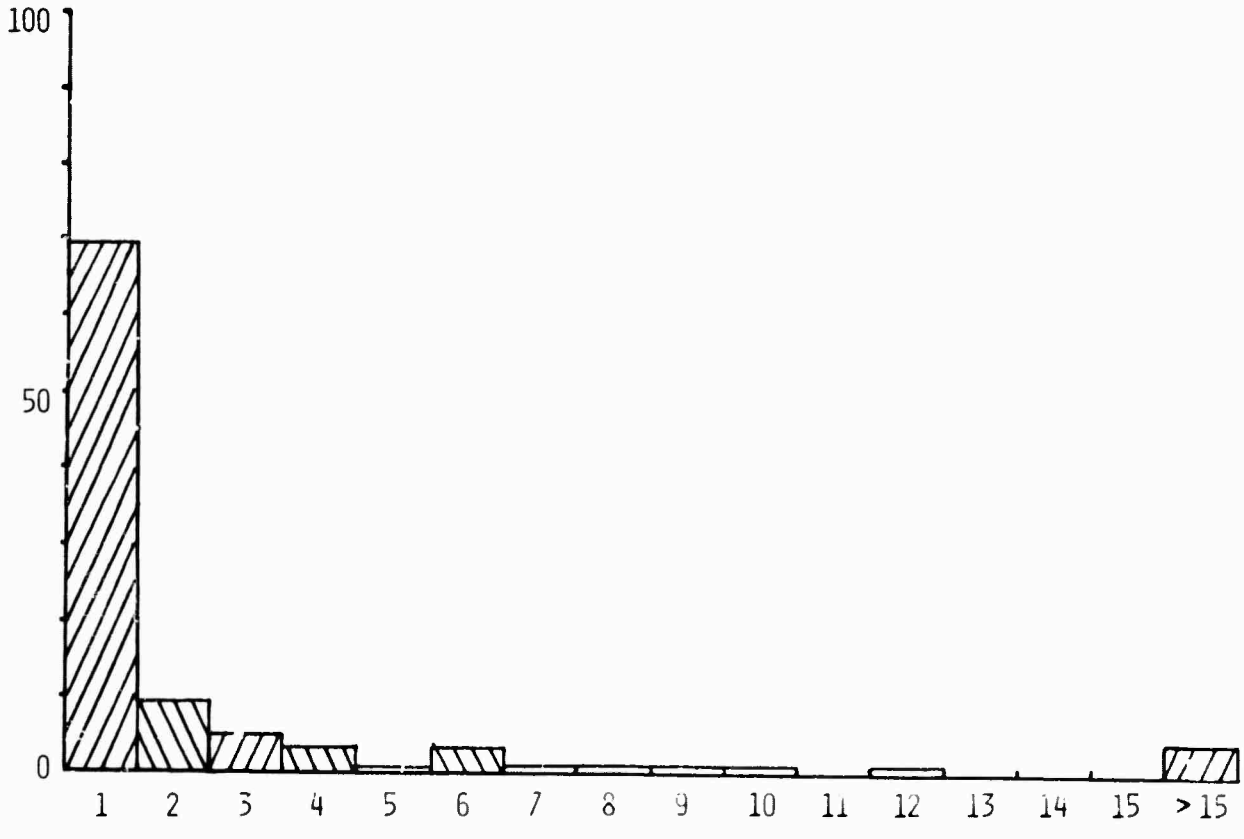


Fig. 13. Position of first correct pronunciation in the initial scan.

Reference

Woods, W.A., M. Bates, G. Brown, B. Bruce, C. Cook, L. Gould, J. Klovstad, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue (1975)
"Speech Understanding Systems, Quarterly Technical Progress Report No. 4," BBN Report No. 3188, Bolt Beranek and Newman Inc., Cambridge, Mass.

C. Shortfall Scoring Strategies for Speech Understanding Control

W. A. Woods

Abstract

This note describes several methods of assigning priority scores to partially developed hypothetical interpretations of a speech utterance to determine which ones to extend further. Several of these methods can be proven to guarantee the discovery of the best matching interpretation of the utterance, under appropriate assumptions about the lexical scoring metric used. All are variations on a method called "shortfall scoring." The method consists of determining a tight upper bound on the score of the best possible sequence of words that could cover the utterance and giving each alternative partial hypothesis a "shortfall score" equal to the amount by which the score for the hypothesis falls below the maximum score for that region.

1) Introduction

In the following discussion, we are assuming a continuous speech understanding system that contains the following parts:

a. A Lexical Retrieval component that can find the k best matching words starting or ending at any given point in the utterance for any number k, and can be recalled to

continue enumerating word matches in decreasing order of goodness at a given Position.

b. A Linguistic Consultant component which given any sequence of words can determine whether that sequence can be parsed as a possible initial, final, or internal subsequence of a syntactically correct and pragmatically appropriate utterance.

c. Appropriate signal processing, acoustic-phonetic and phonological analysis components.

The control problem for such a system is to determine:

a. At which points in the utterance to call the Lexical Retrieval component.

b. What number of words to ask for.

c. When to give subsequences of the results to the Linguistic Consultant.

d. When to recall the Lexical Retrieval component to continue enumerating words at a given point.

This paper will present several control strategies that embody answers to these questions, some of which are guaranteed to find the best matching, linguistically acceptable, sequence of words that covers the utterance, and do so without exhaustively enumerating all the possibilities.

The strategies are all variations on a basic scoring method that I have dubbed "shortfall scoring." The method operates in the context of an island-driven control framework in which selected "seed" word matches are used to form initial partial theories about the possible identity of the utterance, and these partial theories are subsequently

extended by the addition of new word marches to their ends. The essence of the shortfall method is as follows:

a. Use the set of possible boundaries detected by the Acoustic-Phonetic component to partition the utterance into elementary segments the size of a phoneme or smaller.

b. Determine for each such segment the maximum possible contribution to the score of any theory that can be attributed to that segment by any word match.

c. Assign to each partial theory a priority score consisting of the amount by which its accumulated lexical score falls below the maximum possible score for the region that it covers.

d. Use this priority score to determine which partial theory to pursue next.

The method can be applied to other perception tasks, besides speech understanding (e.g., image understanding), by using appropriate analogs of phoneme and phoneme boundary.

We will first present the basic shortfall method, and argue that it finds the best possible interpretation of the utterance under appropriate assumptions about the lexical scoring metric and the characteristics of the Lexical Retrieval component. We will then present a number of variations of this basic method and a discussion of the advantages of the various options. The strategies that we will describe include:

a. Basic shortfall strategy

b. Shortfall plus subsumption shelving.

c. Shortfall plus credit or liability.

d. Shortfall density.

e. Shortfall density plus island collisions.

f. Shortfall density plus ghost words.

g. Shortfall density plus ghost words and chosen direction.

The meanings of the terms "subsumption shelving", "credit", "shortfall density", "ghost words", "chosen direction", and "island collisions" will be taken up in subsequent sections. All but the island collision strategy have been implemented in HWIM and tested. Each option, such as "ghost words," is associated with a different flag that can be enabled or disabled. Thus various combinations of options can be tried. However, only the strategies listed above (with the exception of island collisions) have yet been tried experimentally.

2) The Basic Shortfall Scoring Method

In the next few subsections, we will briefly introduce the basic shortfall method and discuss its completeness (i.e., the degree to which it guarantees to get the best quality interpretation of the utterance). In subsequent sections, we will discuss variations on the basic method.

2.a Assumptions

The shortfall method assumes that word matches are assigned quality scores by a Lexical Matching component, and that theories are assigned scores which are the sums of the scores of the word marches contained in them. If the scores are multiplicative, then the same method can be applied using scores in log space or an analogous method using products can be applied. (Actually any function of the scores of the separate words which is symmetric and non-decreasing will work, but will give a strategy that is somewhat more breadth first and will follow more stray paths before getting to the right one.) The method requires a lexical retrieval algorithm that can enumerate in order the best matching words in the entire utterance and the best matching words adjacent to any given word. (The Lexical Retrieval component in the current HWIM system (see Section II.B.) has such capabilities. It also assumes that words are scored against a segment lattice of input phonetic elements and are given beginning and ending positions which correspond to junctures in this segment lattice.

2.b Basic Shortfall Scoring Procedure

Let t(i) be the time in milliseconds of the ith boundary in the segment lattice, nsegs, the number of segments; and seg(i) be the region of the input utterance

from t(i-1) to t(i) for i from 1 to nsegs. (Note that seg(i) is not necessarily one of the segments in the segment lattice, and could be of length zero if two segment boundaries happened to have the same t(i).)

For a word match from position i to j with score q, we will allocate in some systematic way the total word score q to the segments seg(i+1) ... seg(j) covered b the word match. For this discussion, let us allocate it proportional to the lengths of the segments.

For a given segment lattice, we will determine for each segment seg(i) the maximum score max(i) that can be allocated to that segment by any word match that covers that segment. The score for any word match from i to j will hence be bounded by the sum max(i+1)+ ... +max(j), and the maximum score for any complete theory will be bounded by T = the sum from 1 to nsegs of max(i).

Every partial theory will consist of a contiguous sequence of word matches spanning a region from some boundary i to some boundary j. Each such theory will carry with it two scores m and q, where m is the sum of the max(i) for the segments covered by the sequence and q is the sum of the word scores of the theory. We will assign each theory a priority score p = T - m + q, which can be thought of as the maximum total score T for any theory T minus the shortfall from this ideal to which one is committed by choosing this

particular sequence of words for the region from i to j (i.e., p = T-(m-q). Alternatively, it can be thought of as the estimated best possible future score consisting of the score q which has been achieved for the region already covered plus the best potential score T-m for the region not yet covered (i.e., p = q+(T-m)). Because T-m is an upper bound on the possible score that can be achieved on the region not covered, the priority scores p have the characteristic that they are strictly decreasing.

New theories arise from processing events that link an existing theory with a new word match. The m and q scores of an event and the new theory that it represents are simply the vector sums of the m and q scores of the old theory and the word being added to it. Thus, after assigning an m score to a word match by summing the max numbers for the segments that it covers, the m score of any new theory that includes it is computed by a single addition.

2.c Strategy

The basic control framework in which the shortfall method is applied consists of an initial scan of the utterance to obtain the m best shortfall word matches wm(1) ... wm(m). which are then put in an event queue as initial seed events with priority scores as determined above. In addition, a continuation event, to continue the initial scan

for words with lower scores, is placed in the queue and given a priority score equal to that of the lowest scoring word match, wm(m). Subsequently, the algorithm repeatedly processes the highest priority event until a complete spanning theory is found and processed.

Processing a seed event consists of forming a one-word theory which is given to the linguistic consultant to obtain predictions of adjacent words. These predictions are then given to the Lexical Retrieval component to obtain the best word matches at each end of the theory plus continuation events for finding additional matches. Each word match found by this process results in a word event to add the new match to the theory that proposed it.

Processing a word event is similar to that of a seed event in that a theory is formed and given to the linguistic consultant for evaluation. If the theory is acceptable to the grammar and has not violated any semantic or pragmatic requirements, then new proposals are generated, which in turn generate new word matches and new word events.

2.d Completeness of the Method

Claim:

The first complete spanning theory found by the above process will be the best scoring complete theory that can be found by any strategy.

Proof:

At the time the first complete spanning theory has been processed, every other event on the event queue (including continuation events for finding lower scoring seeds or lower scoring words to add to the ends of islands) will already have fallen low enough in its partial score (q score) that no possible match sequence in the remaining region of the utterance can bring its total score up to that of the spanning theory. Also, the presence of the continuation events in the queue makes the search process complete in the sense that any word in the vocabulary would be enumerated if the process were continued long enough. Thus there is no possible word sequence across the utterance that would not be considered by this search algorithm if it were run sufficiently far. Hence, any complete theory of the utterance will have a shortfall score (m-q) not less than the one of the first complete theory discovered by the above strategy. Since all spanning theories have the same maxscore m, it follows that the first spanning theory also

has the maximum possible quality score (q) of any spanning theory.

Notes:

Note that the process can be continued to obtain the second best complete theory, and so on.

Note also that since the only use of the total maximum T is in the comparison of two priority scores of the form T-m+q, the T's on both sides of the equation cancel. Therefore, using the priority scores m-q instead of T-m+q (and working on the event with the smallest instead of the largest priority score) will result in the same strategy. That is, a scoring strategy that uses only the local shortfall score s = m-q as its priority score can make all the same decisions as the above described algorithm.

2.e Assumptions About the Grammar

The shortfall method works well with almost any type of grammar. It makes no assumptions that the grammar is finite-state as do most Markovian strategies. It does require the linguistic consultant to have a parser (such as the bidirectional ATN parser in the current HWIM system) that can take an arbitrary island fragment in the middle of an utterance and judge whether it is a possible subsequence of an acceptable sentence. In practice, it helps immensely

if the parser can also use the grammar to predict the acceptable words and classes adjacent to the island, and if the Lexical Retrieval component can use such predictions (to constrain its search), but this is not essential to the formal completeness of the algorithm.

2.f Avoiding Duplicate Theories

Note. that in this method, there are many different ways of eventually arriving at the same theory. For example, if we have an island w with a possible word x on the left and a possible word y on the right, then we can first form the theory (xw) and then (xwy) or we can form the theory (wy) and then derive (xwy) from that. Which of these two routes is taken will depend on the scores of the words, but it is quite possible (in fact, likely) that in the course of working toward a complete theory the strategy will arrive at the same subtheory several different times by alternate routes.

For example, assume that for a one-word theory (w), the events x(w) and (w)y have shortfall scores of 5 and 6, respectively. The event x(w) will then be processed. forming a new theory (xw) with shortfall 5. Assume this theory generates events z(xw) with shortfall 9 and (xw)y with shortfall 7 (e.g., w had a shortfall of 4 and x scored 1, y scored 2, and z scored 4). Then we would have three

unprocessed events on the queue:

(w)y 6

(xw)y 7

z(xw) 9

Both of the events leading from (xw) have fallen below the event (w)y, and so we process (w)y resulting in a new theory (wy) with score 6, which has events x(wy) with score 7 and (wy)u with score 8 (i.e., u has score 2). As a result we will have derived two events (xw)y and x(wy) both on the stack with the same score and both of which will create the theory (xwy).

If we do not include checks for the duplication of theories, then we would often get two copies of the same theory. These would forever duplicate the same predictions and theory formations, giving rise to a rapid exponential explosion of the search process. If we include a test each time a theory is formed to determine whether that theory has been formed previously, then we can avoid such an exponential process. In fact, if each time we are about to put an event on the event queue we check the event to see that the set of word matches that it uses is not the same as that of some other event, then we can terminate this duplication before making the entry on the stack and consuming the stack space (and certainly before calling the Linguistic Consultant to check it out and make further

predictions).

The check for duplication among all the events that have been crated can amount to a considerable amount of testing if done in a brute force exhaustive test, although it can be considerably reduced by indexing events by their beginning and end points or other tricks. However, if one can rely on the events being generated in the order determined by the basic shortfall strategy and words being returned from the Lexical Retrieval component in order of increasing shortfall, then the following simple check based only on the word matches at each end of an event can be used to determine whether it is redundant (i.e., will produce the same theory as some event already generated):

> If the new word is at the left end and has the same or greater shortfall as the word at the right end, then this event is redundant.
>
> If the new word is at the right end and has strictly greater shortfall than the word at the left end, then this event is redundant.

The argument for the validity of this test is as follows:

In the search space we are considering, it is possible, without the check for duplication we are considering, to derive a given theory with words w1,w2,...,wk in 2**k-1 different ways -- one corresponding to each of the possible binary derivation trees starting with some one of the wi as a seed, and then successively adding words either to the right or the left end. (Proof -- either w1 or wk was chosen

last, hence there are two ways to derive a string of length k for every possible derivation of a string of length k-1. There is one possible way -- i.e., as a seed -- to derive a string of length 1.) Of all these derivation trees, the first one that will be found is the one that uses the wi with the smallest shortfall as a seed, and at subsequent steps adds the better (in terms of shortfall) of the two words at either end (assume for the moment that no two of the words have exactly the same score). Hence, any derivation that attempts to add a word to one end of an island that has smaller shortfall than the last word added to the other end of the island (or the seed if that is the word at the other end of the island) will be duplicating a theory that has already been derived (or at least already has an event for it on the event queue). In the case of two competing seeds with the same shortfall or words at each end of an island that have the same shortfall, we can arbitrarily assume that the leftmost is the preferred one, which we will permit the algorithm to follow, and we can block the other one. In this case, if we have a word being added to the left end of a theory that has the same shortfall as the word at the right end, then this event is redundant.

Thus a very simple check between the score of the word being added to a theory and the score of the word at the other end of the theory will suffice to eliminate the formation of redundant events (assuming words are returned by Lexical Retrieval in increasing shortfall order).

2.g Fuzzy Word Matches

The above discussion does not explicitly mention the problem of finding the same word in essentially the same place but with slightly different end points and different scores. We have observed this kind of output from the Lexical Retrieval component and indeed find it desirable to know the degree of variation possible in the end points of a word match and the appropriate degradation in score for each. However, it is wasteful to give several different events to the Linguistic Consultant component, all of which are adding word matches to a given theory, which differ only in their endpoints and scores. For this reason, we have introduced a structure that groups together multiple equivalent word matches into a single entity called a fuzzy word match (or "fuzzy" for short), which is given the score of it best member. A theory containing fuzzy word matches actually represents a class of grammatically equivalent theories and carries the score of the best one.

When an event is created to add a word match to a theory containing a fuzzy word match at that end, the score of the event must be computed using a rectified score for the theory which takes into account the best member of the fuzzy that is compatible with the new word (i.e., has boundaries that hook up to the new word and satisfies appropriate phonological word boundary constraints). In general, when several fuzzies are adjacent, the best compatible sequence of members must be chosen, and when the new word match is itself a fuzzy, the best combination of one of its members with a corresponding rectified score for the theory must be taken. The event is thus given the score of the best of the grammatically equivalent, non-fuzzy events for which it stands.

If word matches returned by the Lexical Retrieval component are grouped into fuzzy matches wnenever possible, and word events are given appropriately rectified scores, then the above completeness result still holds (i.e., the first complete theory processed will be the best). The only difference (aside from the elimination of separate processing for grammatically equivalent theories) will be that certain word events will be formed earlier than they otherwise would have. However, these events will still be placed on the queue with the correct score so that they will reach the top and be processed in exactly the same order as they would in the strategy without fuzzies.

2.h Discussion

The method just described is similar in some respects to the well-known branch and bound technique, except for the characteristic that the same partial interpretation may be reached by many different paths, and the fact that the space of possible solutions is determined by a grammar rather than a set of assignments of values to a predetermined set of variables. It can be more completely modeled as an example of the A* algorithm of Hart, Nilsson & Raphael [1968] for finding the shortest path through a graph, where, in this case, the nodes in the graph are partial interpretations of the utterance, and the connections in the graph correspond to the seed and notice events. It is simpler than the general A* algorithm, however, in that we are looking for the best scoring node, and we are not interested in scores of paths leading to that node. The simple argument given previously suffices to show the completeness of the shortfall method, whereas the general A* algorithm is more complicated.

The advantage of the shortfall method over using the quality scores alone as priority scores is that as a theory is extended its shortfall score can only get worse (larger shortfalls). This guarantees that once the score of a theory falls below the maxscore profile by a given amount, no further refinement of the theory can have a better

overall shortfall. This is sufficient to guarantee that the first grammatically spanning theory will have the lowest possible shortfall and the highest possible quality score.

Measuring the shortfall from any profile that is an upper bound of the final score would be sufficient to assure this theoretical completeness. However, the tightness of the upper bound affects the number of events tried and partial theories created in the search for a successful interpretation (i.e., the "breadth" of the search). By assigning the upper bound as a maximum segment score profile determined by allocated shares of actual word match scores, a fairly tight upper bound is achieved.

A further effect of scoring the shortfall from the maxscore profile is that the score differences in different parts of the utterance are effectively leveled out so that events in a region of the utterance where there are not very good scoring words can hold their own against alternative interpretations in regions where there are high scoring words. This promotes the refocusing of attention from a region where there may happen to be high scoring accidental word matches to an event whose word match quality may not be as great, but is the best match in its region. Notice in the example in section 2.f, that the decision to add x to the left of w based on an initially better score is later queued lower after finding the low scoring additional event

z(xw) in favor of the event (w)y which yields (wy)u scoring better than z(xw). Thus, this ability to back up to an event that initially didn't score as well as some other event, but which can be pushed farther with an aggregate better score is automatically handled by the shortfall scoring method. Thus an apparently satisfactory and intuitively reasonable strategy for focusing of attention is emerging from the same strategy that guarantees to get the best scoring theory first.

When using the shortfall method for understanding an utterance, the overwhelming tendency is for an event adding a new word to an island to pick up additional shortfall and fall some distance down in the queue. The result is that other events are processed before any additional work is done on that island. (Occasionally, the new word is the best word in its region and buys no additional shortfall, but this is a rarity.) The distance that this new event falls down the queue is determined by the amount of additional shortfall that it has just picked up and the shortfalls of the events that are competing with it on the queue. This distance directly affects the degree of "depth-first" vs. "breadth-first" processing done by the algorithm. If the new word scores well, the event falls only slightly, few, if any, alternate events are processed before it, and the algorithm is relatively depth first. If the new word scores badly, the event falls further down the

queue, many more alternative events have priority over it and the algorithm is more breadth first.

The above characterization is only an intuitive approximation, since the actual number of events processed before the new event is considered depends on the number of new events that will be generated by the intervening events that will still score higher than this one. In some cases, the number of such events can be extensive. The general effect, however, is that the shortfall scoring method provides a dynamically varying combination of depth-first and breadth-first search which is determined by the relative qualities of the events that are in competition.

3) Deviations from the Ideal Shortfall Method

In the actual implementation of the shortfall strategy, we have made a number of departures from the theoretical method for either expediency or practical efficiency. One such modification is the omission of continuation events, on which the theoretical completeness of the algorithm depends. These events provide for indefinitely continued enumeration of seed events or matches of proposed words of successively lower quality. In practice, however, one cannot afford to run the algorithm indefinitely, even if one is guaranteed eventually to get an answer. We have implemented the algorithm without continuation events and instead attempt to

provide for a sufficiently large number of words to be returned by the initial scan and by the requests for proposed words. As long as there is still a seed event on the event queue, and each island still has pending events on the queue, one is guaranteed to be following the ideal algorithm. Even when this guarantee is lost, one is still very likely to find the best possible interpretation in those cases where he could have afforded to run the theoretical algorithm to completion.

A second modification to the basic algorithm is the elimination of certain words and syntactic categories as possible seed words. We have previously noted [Woods, 1975] that the small function words in English tend to match well accidentally at many points in an utterance. These words make poor seed theories even when they match well, since their chance of being correct is relatively small. When permitted, spurious events for these seed theories tend to clutter up the top of the queue until the events that they spawn fall far enough down to be out of consideration. The elimination of these words as seeds does not affect the possibility of obtaining a given spanning theory as long as that spanning theory does not consist entirely of function words.

It is possible, of course, to have utterances in a dialog that consist entirely of function words, if one's definition of function word is sufficiently broad (e.g., "What is it?", "Yes", or "It's O.K."), but if one constrains that definition so that every grammatical utterance has at least one non-function word, then this restriction will not lose any utterances. (For example, neither "what", "Yes", nor "O.K." are considered bad seeds in our system.) As a variation, one can make the notion of bad seed word context-dependent. For example, we have implemented a variation of this that does not permit number words as seeds except in a discourse context where a possible answer might consist entirely of number words.

A third and fairly major deviation from the ideal algorithm is a provision to rank spanning events (i.e., events that completely cover the utterance) higher than non-spanning events regardless of score. (We have implemented this on a flag (QUICKFLAG) so that it can be turned on or off.) This feature was implemented to speed up the discovery of spanning events in utterances where the last word added to form a spanning theory has a fairly low score. The resulting score of such an event may place it sufficiently low in the queue that many intervening events have to be considered before it bubbles to the top of the queue. The QUICKFLAG feature introduces substantial risk that the spanning event one finds is not the best one, since

an event that would otherwise be above it might indeed extend to a spanning event with better score. The guarantee could again be restored by providing special processing after the first spanning event was found to verify that there can be no better spanning event, but at least for the shortfall density method (to be described shortly), no case of a premature spanning event has yet been encountered. Since our testing so far has not been extensive, this is possibly just a lucky accident, but it is made plausible by the fact that a theory that can be extended to a complete spanning theory by the addition of one more word, and is the best such theory, is very likely to be correct. It does occasionally happen that several spanning events are discovered at the same time, due to several alternative words being noticed that could complete the theory. In this case, however, these alternatives are ranked according to their shortfall score, so the first one of them that is successful is the best.

Even with the slight theoretical risk of a less-than-best interpretation of an utterance, the number of events processed in understanding a typical utterance is sufficiently reduced by the QUICKFLAG feature, that it is a well-justified practical modification to the theoretical method.

The fourth and most serious deviation from the theoretical algorithm, true of our current implementation not by design but by historical legacy, is that the word matches returned by the Lexical Retrieval component from both the initial scan and subsequent predictions are returned in order of decreasing score rather than in order of increasing shortfall. This results in the possibility of many seeds being found in regions where words score well, with few, if any, seeds being found in regions where the best scoring matches are not very high in quality. For the basic shortfall scoring strategy, this is not a severe difficulty, as long as the chances of some word of the correct interpretation being found as a seed are virtually certain. However, for some of the variations on the basic method, such as the use of subsumption shelving (to be described next), it results in a significant loss of the guarantee of finding the best interpretation and sometimes an increase in the number of events processed before finding a correct spanning theory.

Words are returned in decreasing order of score by the current Lexical Retrieval component because this was a desirable feature for previous control strategies, and we have not yet had time to build a Lexical Retrieval component that is specifically matched to the shortfall method. This mismatch can to a large degree be offset by writing a more complicated initial scan routine which assures an even

spread of word matches across the utterance in spite of lower scores in some regions. However, our results from using the shortfall scoring method have been fairly encouraging, even with the enumeration of initial seed words in decreasing score order. By setting the initial scan threshold low enough and requesting a sufficiently large number of words in the initial scan, we achieve a close approximation of the ideal ordering in the seed events at the top of the event queue, and only further down in the queue do we face the possibility that there are word matches that have not yet been found whose shortfall, when computed, would place them at that point in the queue.

4) Variations on the Basic Shortfall Method

4.a Subsumption Shelving

The basic idea of subsumption shelving is that if events are enumerated in strictly increasing shortfall order, then at the moment that a given event reaches the top of the queue, no event on the queue which it subsumes can lead to a better spanning theory unless there is a sufficiently good seed word somewhere outside its region to make up for the additional shortfall of these subsumed events. In that case, however, the spanning theory in question can be derived from this other good seed. Consequently, such subsumed events can be placed on a shelf

and removed from the event queue. They cannot be pruned from consideration entirely, however, since the subsuming event that caused them to be shelved may not be extendable to a spanning theory. That is, every theory derived from the shelving theory may be rejected by the Linguistic Consultant due to internal inconsistencies or incompatibility between the end points of the hypothesized interpretation and those of the actual utterance. In this case, the decision to shelve the subsumed events in favor of the shelving theory can be seen to be a mistake, and the shelved events need to be restored to the event queue for consideration.

The argument justifying subsumption shelving is that since events are enumerated in increasing shortfall order, once an event reaches the top of the event queue and is processed, any event that could cover that region with a better shortfall will have already been processed. Hence, subsumed events that have strictly larger shortfall than their subsuming event can only lead to poorer ways of covering that region. Furthermore, if the subsuming event can be eventually extended to a complete spanning theory (call that potential spanning theory, ALPHA), then any spanning theory that might be derived from one of the subsumed events can only score better overall than theory ALPHA if there is some word match outside the region of the subsuming theory that has less shortfall than ALPHA and

hence will be processed as a seed event before ALPHA is completed. Thus whenever a shelved evert can lead to the best spanning theory, there will always be a seed word match somewhere else from which the same spanning theory can be derived.

The only flaw in the above argument is that the other seed word, on which finding the correct spanning theory depends, may itself have been shelved by some other event. As a result, some lesser quality spanning theory may be discovered before all of the events derivable from the shelving events have been rejected and one or the other of the critical events is unshelved. When words returned from Lexical Retrieval are <u>not</u> enumerated in shortfall order, the enumeration of this other seed word may be delayed -- again risking a spurious spanning theory. Moreover, if continuation events have been omitted, the needed seed may not be enumerated at all. Thus, subsumption shelving introduces a fairly high risk of not finding the best possible interpretation and is sensitive to the order of seed enumeration. In addition, while it significantly reduces the number of events processed for many utterances, for some other ones that number is increased, so its ultimate value is questionable. Nevertheless, the method is described here for its interest and for comparison with other, later variations.

There are two methods of implementing the unshelving of events in order to insure that the best spanning theory is not eliminated from consideration entirely. Both of these have been implemented and can be chosen by the setting of flags in the system. The first one simply consists of restoring all of the shelved events whenever the event queue becomes empty. When running in this mode, the event queue tends to get shorter and shorter, as more and more events get subsumed and shelved. If a complete spanning theory is found before the event queue becomes empty, then that is the chosen theory (with some risk that a better spanning theory may have been derivable from some shelved event). If one is not found, all shelved events are restored and the process continues. Events are reshelved when they become subsumed again by new events reaching the top of the queue, and this process is continued until some spanning theory is discovered.

The second method of unshelving is more complicated. It consists of remembering the shelving theory with the batch of events that it shelved, and detecting when the event queue no longer contains any events derived from it. This process is made tractable by gathering up the live theories (i.e., theories that have descendants in the event queue) and testing whether there are shelved events whose shelving theory is not live. This is done whenever an event fails to produce further descendants (and hence may be the

last descendant of one or more theories). At that point, all the events shelved by dead theories are restored as long as they would not have been shelved by some other live theory.

The major disadvantage of the first unshelving method over the second is that one may have to wait longer than with the second method for some events to be unshelved. This is true because they will not be unshelved until the event queue empties instead of being unshelved as soon as the last descendant of their shelving theory dies. On the other hand, this method guarantees that all shelved events will be unshelved periodically if no spanning event is found. In the second method, especially with continuation events, it is possible for some theory to continue to have active descendants until the vocabulary is exhausted, thus making some shelved events for all practical purposes unrecoverable.

In summary, the subsumption shelving feature is an option that can in many cases significantly reduce the number of events that have to be considered in order to find a spanning utterance. However, it can occasionally increase that number and also runs some risk of finding a less than best spanning event first.

4.b Credit and Liability

One way of looking at shortfall scoring is to consider the expression T-m+q of our original derivation to be an estimate of the final score of a spanning event derived from the current event. That is, we have already achieved a score q for the current event, and we can be sure that the best possible score that can be achieved on the remaining uncovered portion of the utterance is not greater than T-m (i.e., the area under the maxscore profile for the portion of the utterance not covered by the current theory). This is clearly a conservative bound which will hardly ever be achieved, particularly when we are working on events in increasing shortfall order and can be assured that when we process an event with a given shortfall s, any word matches elsewhere in the utterance with no shortfall (or any shortfall less than s) would already have been processed. If we could derive a much tighter bound on the ultimate score that can be achieved by a given event, but one that is still an upper bound, then we could eliminate a great deal of breadth in our search process and proceed much more directly to a spanning interpretation.

One way to do this is to realize that after the algorithm has been running for some time, and some number of events have come to the top of the queue and been processed, we now know something more about the best possible shortfall

in the regions of the utterance that have been covered by those events. When an event covering a region from i to j has become the best event on the queue and is processed, we can be sure that any other interpretation that covers that region with a smaller shortfall will have already been found. If we remember the best quality score that has been achieved for any region of the utterance, or the best shortfall that has been achieved on each region, then when we are considering the priority scoring of an event, instead of assuming the shortfall outside the event to be zero, we can add to the shortfall already achieved the best shortfall that it could possibly expect to achieve on the remaining uncovered portion of the utterance. This is estimated from the shortfalls that have already been found by other events in these regions. We call this extra shortfall the liability of the event. It is a minimum for the future shortfall which will have to be bought by any theory that may be derived from the event.

Ranking events with their shortfall plus the liability expected on the uncovered regions still provides a lower bound on the possible shortfall of a potential spanning event, and hence constitutes a complete strategy. The computation of the liability for a given region at a given point in the algorithm is somewhat complicated, requiring one to remember the score of the best theory that has been processed for each region, to consider all of the sequences

of such best partial theories that fall within the region for which the liability is to be computed (and which are not properly subsumed by some other such sequence), and to determine the smallest shortfall of any such sequence. This computation is fairly expensive in computer time and can only be made tractable with careful programming techniques, but appears in practice to be worth the effort by causing spurious, locally good events to be placed at a more appropriate rank in the event queue.

When running the shortfall plus liability strategy, it is characteristic for events in the queue to be rescored and fall lower in the queue as more is learned about the liability which they will have to eventually accept. This causes the correct events to bubble to the top of the queue much more rapidly.

A slightly different variation, which is similar to the liability strategy, is to give credit to an event for the shortfall in the regions within the event which any other theory that eventually covers the region will also have to accept. This method is the dual of the liability method, in that the same computation that determines the liability of a region is applied to the region covered by the event instead of the regions not covered, and is subtracted from its shortfall instead of added to it. The difference between the two strategies is somewhat like that between raising a

bridge and lowering the river, but the effects are not quite the same, since what is credit for one event is not necessarily liability for another (e.g., when the two events in question overlap). In the credit strategy, we are essentially replacing the maxscore profile for selected regions of the utterance with the best possible score that has been achieved by a partial theory covering that region. This reflects the fact that the original maxscore for the region was a maximum that might not be achievable and attempts to compensate with a revised estimate of the best that can be achieved for the region. Shortfall plus credit is not necessarily a complete strategy, but it appears to be a fairly effective one in practice.

4.c Shortfall Density and Island Collisions

Shortfall density scoring consists of using the shortfall of an event divided by the duration of the region covered by the event as the priority score. One way to view this strategy is to consider again the task of estimating the expected shortfall to be achieved in the region not covered by an event and consider estimating this liability as a direct extension of the same shortfall per millisecond as has already been achieved by the event --i.e., add to the current shortfall a liability consisting of the shortfall density of the current event times the duration of the region not covered by the event. Since the resulting total

shortfall is just the shortfall density of the event times the total duration of the utterance, and the total duration of the utterance is a constant, we can compare only the shortfall densities and achieve the same decisions. (Actually, in the implementation we carry around both the shortfall and the duration and compare short(a)*dur(b) with short(b)*dur(a) to rank two events a and b, in order to avoid division and floating point numbers and to work with small integers. This makes the computation of the new density from an old event and a new word consist merely of two adds rather than a weighted average.)

When we think of the shortfall density as an extrapolation of the shortfall already bought by an event into the region not covered by the event, we are clearly no longer obtaining an upper bound on the possible future score of an event and the previous proof of completeness for the shortfall method no longer applies. In particular, whereas the shortfall is a monotonically increasing function as an island grows, the shortfall density can get larger when a bad word is picked up and then get smaller as the island grows and picks up better words, averaging the shortfall of the bad word over a larger duration. Thus it is not true that the shortfall density of descendants of an event must be greater than that of the event itself.

However, when combined with an island collision feature that allows one to combine together in one step the word lists of two different events that are predicting the same word from opposite sides, the shortfall density method is a complete strategy, and even without the island collision feature it is a very effective control strategy.

To prove the completeness of the shortfall density plus island collision strategy, we must use a more complicated argument than for the basic shortfall strategy. The argument depends on the ability to derive the same theory in different ways from different seeds.

Proof:

For a given event on the event queue ordered by shortfall density, consider two cases: (1) the extrapolated shortfall density is indeed a lower bound on the ultimate shortfall density of any descendant of the event, and (2) it is not. In the first case, the event is appropriately scored in the event queue for the shortfall plus liability strategy. In the second case, in order for the final shortfall density of some spanning descendant of the event to be less than that of the event itself, there must be a word or words of lower shortfall density which will be combined with it to lower its density. But in this case, each of those words will appear as a seed in the queue ranked better than the event in question. Moreover, the

shortfall density of all the additional words combined must be less than the density of the event in question in order for the spanning event to have a lower shortfall density, and hence the better scored seeds can be expanded right up to the boundaries of the event in question before this event would come to the top of the queue.

Clearly the spanning event is derivable from either of these two events, and so the event in question is not essential to the discovery of the spanning event desired. Thus type 2 events are not necessary for the discovery of any spanning theory with lower shortfall density than their own, and the extrapolated shortfall density for such events is indeed a lower bound on any of the spanning events that we need to derive from them. Hence, when the first complete spanning theory comes to the top of the event queue, all of the type 1 events that remain in the queue can only lead to events with higher shortfall density, and any spanning event derivable from one of the type 2 events will either have a higher shortfall density than that event, or will be derivable from some type 1 event earlier than that event in the queue. The only thing that keeps this algorithm from being complete as it stands is that this earlier type 1 event may in fact have already been processed and the derivation leading from it may have temporarily gotten a higher shortfall density due to picking up a higher density word. Then before it has had the chance to pick up

additional words and bring its shortfall density back down, some other spanning theory may be found. Thus it is possible for an event that could lead to a better overall density to remain untried due to a locally higher density than that of some spurious spanning event that is found first.

Adding an island collision feature eliminates this chance for the best spanning theory to be missed. Recall that each type 2 event has a corresponding type 1 event which has better shortfall density and abuts the type 2 event. If the type 1 event is able to pick up successive words from the type 2 event without its shortfall density going above that of the final spanning theory derivable from it (i.e., without becoming a type 2 event), then no island collision is necessary, and the combination of the two word lists will be accomplished one word at a time before any spanning theory of higher shortfall density can be found. If, however, before the type 1 event has succeeded in picking up all of the words from the type 2 event, it picks up a word that has sufficiently high density to cause its density to go above that of the final spanning event, then the remaining words in the type 2 event, taken together must have a lower shortfall density than the event we have just grown and must therefore have already been formed into a theory and will have noticed it is same word from the opposite side. Hence, the island collision strategy will

permit the word match lists for these two events to be combined at a stage in the algorithm where no events with shortfall densities greater than that of the combined two islands have yet been processed. This permits the derivation of any spanning theory by a sequence of events, including collision events, whose shortfall densities are monotonically increasing and therefore at the time when the first complete spanning theory comes to the top of the queue, any hypothetical spanning event that might have had a lower density would have had to be already derived. Hence, the shortfall density plus island collisions strategy can be guaranteed to be complete.

4.d Ghost Words

The ghost words option is a feature that can be added to either the shortfall or the shortfall density strategy, and does not affect the completeness of the strategy to which it is added. Every time an event is given to the linguistic consultant for evaluation, proposals are made on both sides of the resulting island (unless the island is already against one end of the utterance). Although events can only add one word at a time to the island, and this must be at one end or the other, eventually a word will have to be added to the other end, and that word cannot score better than the best word that was found at that end the first time. The ghost words feature consists of remembering with

each event the list of words found by the Lexical Retrieval component at the other end and scoring the event using the best of the ghost words as well as the words in the event proper. The result is that bad partial interpretations tend to get bad twice as fast, since they have essentially a one-word look ahead at the other end that comes free from the linguistic consultant each time an event is processed. On the other hand, an event that has a good word match at the other end gets credit for it early so that it gets processed sooner. The ghost words feature, thus, is an accelerator that causes extraneous events to fall faster down the event queue and allow the desired events to rise to the top faster. Experimental use of this feature has shown it to be very effective in reducing the number of events that must be processed to find the best spanning event.

4.e Choosing a Chosen Direction

When a theory is evaluated by the linguistic consultant, predictions are made at both ends of the island. When one of the events resulting from these predictions is later processed, adding a new word to one end of the island, the predictions at the other end of the new island will be a subset of the predictions previously made at that end of the old island. In general, words noticed by this new island at that end will also have been be noticed by the old island, and if the score of the new island is slightly worse than

that of the old island (the normal situation), then the strategy will tend to revert to the old island to try picking up a word at the other end. This leads to a rather frustrating derivation of a given theory by first enumerating a large number of different subsequences of its final word sequence. (For example, to derive a theory (abcd), one might first start with the seed (b), then add a to get (ab), then go back to (b) to get (bc) then go back to (ab) and add c to get (abc), then go back to (bc) and notice a, but also notice that (abc) has already been made and then go back to (abc) to pick up the d.)

Since any eventual spanning theory must eventually pick some word at each end of the island, one could arbitrarily pick either direction and decide to work only in that direction until the end of the utterance is encountered, and only then begin to consider events in the other direction. This would essentially eliminate the duplication described above, but could cause the algorithm to work into a region of the utterance where the correct word did not score very well without the benefit of additional syntactic support that could have been obtained by extending the island further in the other direction for a while.

Without sufficient syntactic constraint at the chosen end, there may be too many acceptable words that score fairly well for the correct poorly scoring word to occur

within a given number of best words at that end. By working on the other end, one may tighten that constraint and enable the desired word to appear, although it can never cause a better scoring word to appear than those that appeared for the shorter island.

The CHOOSEDIR flag in the current system causes the algorithm to pick a preferred or "chosen" direction for a given theory as the direction of the best scoring event that extends that theory, and to mark the events going in the other direction from that theory so that they can only be used for making tighter predictions for words at the chosen end. This is accomplished by blocking from consideration any notices for one of the ghost words at the inactive end of an event if that event is going counter to the chosen direction. This blocking, alone, eliminates a significant number of redundant generations of different ways to get to the same thing. An even greater improvement is obtained by rescoring the events that are going counter to the chosen direction by using the worst ghost at the other end rather than the best ghost. Since only word matches that score worse than any of the ghosts at that end will be permitted by these events, this is a much better estimate of the potential score of any spanning theories that might result from these events, and is still a lower bound, so the completeness is not affected.

The effect of rescoring the events in the non-chosen direction using the worst ghost is that, in most cases, these events fall so low in the event queue as to be totally out of consideration. Only in those cases where there was little syntactic constraint in the chosen direction and the worst matching word at that point was still quite good, do these events stay in contention, and in those cases, the use of the worst ghost score provides the appropriate ranking of these events in the event queue.

5) Empirical Comparison of the Different Strategies

We have not yet been able to do a systematic study of all of the different variations of the shortfall method on a large class of utterances. However, the strategies have each been tested on at least a small number of utterances and the various shortfall density versions have been tested on a somewhat larger number. The variations have been presented here more or less in the order of their discovery, and reflect a generally improving performance trend. Shortfall plus liability is clearly superior to shortfall alone, shortfall density is superior to shortfall alone and to shortfall plus liability, even without the island collisions that are required to make the method formally complete.

We have not yet encountered, to my knowledge, a case where the correct interpretation was shut out by a spurious spanning interpretation of higher shortfall density while it was temporarily high in density due to consuming a bad word match. We have, however, encountered cases where island collisions would have enabled a correct spanning event to be found much sooner than it could be found without them, and where without them the system ran out of space or time before finding any interpretation at all.

The use of ghost words and the chosen direction feature add definite benefit to the shortfall density method, and presumably do also to the basic shortfall method, although we have not run such tests.

The best method, then, at the time of the report, appears to be the shortfall density strategy with ghost words and chosen direction. It appears that other variations on the basic shortfall method may well make further improvements, and some further variations are currently under consideration.

The best shortfall density strategy, so far (using ghost words and chosen direction), is currently finding the correct interpretation of many utterances with a virtual guarantee that they are the best interpretations, after fewer than 50 events have been processed. In some of these cases, our original island-driven strategy, based on quality

scoring alone, which had no such guarantee, ran to over 90 events or ran out of space without finding an interpretation. On the other hand, the shortfall algorithms tend to take a minimum of almost 20 or so events to find most interpretations, while the old island-driven strategy would find many of its successes in only 8 or 10 events. On the whole, however, the shortfall density measure seems to be far superior to basic quality scoring for determining the priority of events in the event queue, especially for the more difficult utterances.

References

Hart, P., N Nilsson, and B. Raphael (1968)
"A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Sys. Sci. Cybernetics, July, Vol. SSC-4, No. 2, pp. 100-107.

Woods, W.A. (1975)
"Motivation and Overview of BBN SPEECHLIS: An Experimental Prototype for Speech Understanding Research," IEEE Transactions on Acoustics, Speech and Signal Processing, Special Issue, February, Vol. ASSP-23, No. 1.